软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, [doi: 10.13328/j.cnki.jos.000000]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

面向跨地理区域联盟链的事务处理技术*

彭泽顺[1], 韩智博[1], 张岩峰[1], 李晓华[1], 于明鹤[2], 范吉立[3], 于戈[1]

[1](东北大学 计算机科学与工程学院,辽宁 沈阳 110169)

[2](东北大学 软件学院,辽宁 沈阳 110169)

[3](东北大学 信息化建设与网络安全办公室,辽宁 沈阳 110819)

通讯作者: 张岩峰, E-mail: zhangyf@mail.neu.edu.cn

摘 要: 跨地理区域联盟链通过将节点分散部署在多个数据中心, 利用区块链的去中心化、不可篡改和可溯源特性, 为电子商务、供应链管理和金融等大规模应用提供支持. 然而, 传统联盟链在大规模部署时面临着性能、扩展性和弹性方面的挑战. 现有区块链在共识算法、并发控制和账本分片上提出了多种方案解决上述挑战. 首先, 本文分别基于网络拓扑结构、主节点数量和网络模型对共识算法进行分类, 并探讨了共识时的不同通信优化策略. 其次, 本文讨论了乐观并发控制、依赖图、确定性并发控制和无协调一致性在跨地理区域场景中的优缺点. 接下来, 本文对区块链跨分片提交协议进行分类, 并分析它们的跨域协调开销. 最后, 本文指出现有跨域联盟链的技术挑战并给出未来的研究方向.

关键词: 联盟区块链;地理分布;共识算法;事务处理;区块链分片

中图法分类号: TP311

中文引用格式: 彭泽顺, 韩智博, 张岩峰, 李晓华, 于明鹤, 范吉立, 于戈.面向跨地理区域联盟链的事务处理技术.软件学报,20XX,XX(X). http://www.jos.org.cn/XXXX-XXXX/XXXX.htm

英文引用格式: Peng ZS, Han ZB, Zhang YF, Li XH, Yu MH, Fan JL, Yu G. Transaction Processing Technologies for Geo-Distributed Permissioned Blockchains. Ruan Jian Xue Bao/Journal of Software, 20XX (in Chinese). http://www.jos.org.cn/XXXX-XXXX/XXXX.htm

Transaction Processing Technologies for Geo-Distributed Permissioned Blockchains

PENG Ze-Shun[1], HAN Zhi-Bo[1], ZHANG Yan-Feng[1], LI Xiao-Hua[1], YU Ming-He[2], FAN Ji-Li[3], YU Ge[1]

[1](School of Computer Science and Engineering, Northeastern University, Shenyang 110169, China)

[2](Software College, Northeastern University, Shenyang 110169, China)

[3](Office of Informatization and Network Security, Northeastern University, Shenyang 110819, China)

Abstract: Geo-distributed permissioned blockchains leverage the characteristics of decentralization, immutability, and traceability to support large-scale applications such as e-commerce, supply chain management, and finance by scattering nodes across multiple data centers. However, traditional permissioned blockchains face performance, scalability, and resilience challenges in large-scale deployment. Existing blockchains have proposed various schemes to address the challenges above in consensus algorithms, concurrency control, and ledger sharding. In this paper, we first categorize consensus algorithms based on network topology, the number of primary nodes, and network models and explore different communication optimization strategies during consensus. Second, we discuss the advantages and disadvantages of concurrency controls, such as optimistic, dependency graphs, deterministic, and CRDT in geo-distributed scenarios. Next, we categorize cross-shard commit protocols for blockchain and analyze their cross-region coordination overheads. Finally, we highlight the technical challenges in geo-distributed blockchains and suggest future research directions.

Key words: permissioned blockchain; geo-distributed deployment; consensus; transaction processing; blockchain sharding

* 基金项目: 国家重点研发计划 (2023YFB4503601), 国家自然科学基金(62372097, 62072086); 国家社会科学基金(21&ZD124); 中央高校基本科研业务费(N2116005, N2116008, N2416003)

收稿时间: 2024-04-22; 修改时间: 2024-08-22; 采用时间: 2025-02-19

区块链作为去中心化的复制账本系统, 提供一种不可篡改、可溯源、可验证的安全执行方案, 已经在跨国金融[1-4]、商务[5, 6]、物联网[5, 7-9]、医疗[10-12]、交通[13-15]等重要领域得到越来越多的应用. 区块链分为公有链、联盟链和私有链 3 种类型. 其中联盟链是一种具有准入机制的区块链技术架构, 具有高性能、安全隐私性高、以及监管友好等特点, 实际应用最为广泛. 在经济全球化发展大潮中, 国内外联盟链应用对于跨地理区域的需求呈现上升趋势. 2022 年发布的 “全球区块链产业图谱报告”[16]中指出, 联盟链应具备高可扩展性, 满足跨区域、跨城市、跨国家的组网要求. 实际应用中, 供应链管理[17]、跨境电子商务[18]、跨境支付[19]、边缘计算[20, 21]等大规模应用都需要跨域联盟链系统提供安全可信的执行平台.

在全球经济场景下, 用户之间的地理距离可能非常遥远, 他们遍布在各大洲的不同国家和地区. 为了保证系统的性能、扩展性和容忍副本失效, 并应对地区监管政策, 全球应用需要将副本分布在多个数据中心[22], 并控制数据的存放位置. 用户可以访问较近的副本使用服务, 避免跨国或跨洲导致的高延迟. 例如, 华为在全球 30 个区域建立了 88 个可用区, 覆盖全球 180 个国家和地区, 全球任何地点的用户可在 100ms 内接入到最近的数据中心[23].

跨域联盟链通常部署在多个数据中心, 其基础设施有如下特点: 一是网络异构, 数据中心内的节点由高速网络相连, 而不同数据中心的节点通过公网连接. 域内带宽比跨域高几个数量级, 同时延迟低几个数量级. 例如, 阿里云数据中心内带宽和延迟可达 40Gbps 与 0.2 毫秒, 而跨域带宽和延迟仅为 100Mbps 与几十至数百毫秒[24]; 二是计算资源异构, 不同数据中心处理效率和存储效率不同. 例如, 企业在不同大洲的用户体量不同, 为节省开销, 不同数据中心部署的节点配置也不同; 三是工作负载异构, 不同区域面向的用户和和访问的服务会产生不同工作负载, 不同负载的性质会影响性能[22]. 例如, 金融应用负载需要可串行化隔离级别, 副本间必须保证线性一致性或顺序一致性. 供应链、边缘计算等应用仅需要读已提交或可重复读等较弱的隔离级别, 副本间满足最终一致性即可[25].

当前的联盟区块链适用于特定组织或者企业, 部署规模和应用不如公有链. 这些企业通常会在其全球数据中心和云基础设施中部署节点, 以支持跨区域的业务运营. 在跨地理区域范围上, 已有研究[26]将联盟链节点部署在不同的大洲. 在大规模跨域网络上, FISCO-BCOS[27]部署了 100 余个节点来评估真实的企业应用场景, 并且其还测试了跨境协作等复杂场景, 在实际应用上提供了良好的扩展性. 除此之外, IBM、微软、阿里云等云服务提供商推出了 BaaS (blockchain as a service) , 将区块链系统作为服务嵌入到云计算平台上, 利用云服务基础设施的部署和管理来扩大区块链的地域范围, 满足企业的大范围大规模业务需求.

因此, 与节点数量多 (如以太坊包含数万全节点), 节点间网络环境复杂 (任何人都可以加入) 的公有链不同, 跨域联盟链的参与者通常为企业, 组织和政府机构, 节点数量较少 (几十至数百), 且节点间的数据中心内和跨数据中心的网络较稳定, 因此其性能远高于公有链系统. 但要支持大规模跨域分布式应用, 联盟链必须克服在性能、扩展性以及容忍副本失效方面面临的难题. 考虑到不同场合下表述习惯, 本文也将区块链系统中的交易称为事务, 这两种说法会同时使用, 其含义相同. 本文通过对相关研究文献进行深入分析, 对这三点指标进行了归纳总结, 指出了传统联盟链跨域部署时会遇到的问题:

1) 性能方面: 大规模分布式应用需要高吞吐率和低延迟. 例如, Visa、支付宝等金融应用需要在同时保证几万到几十万的吞吐量和几十毫秒的延迟[28]. 然而, 由于缺少或不合适的分片策略, 数次跨域协调显著增加交易提交延迟[29]; 由于共识协议并未专门为基础设施做针对性设计, 跨域高通信开销降低了吞吐率[30]; 由于系统并未采用并发控制来执行交易或并发控制跨域场景下性能较差, 吞吐量较低.

2) 扩展性方面: 为满足在全球范围内大规模部署, 系统需要确保可扩展到数百数千个节点; 然而, 主流共识算法采用一主多备架构, 只有主节点支持写操作, 其他节点只支持读操作, 写操作的可扩展性差[31]; 除此之外, 不合适的分片策略在影响性能的同时也会降低扩展能力.

3) 可用性方面: 可用性指跨域联盟链能容忍节点和数据中心宕机的能力. 区块链系统通过将数据在多个数据中心复制保证可用性. 这样在部分节点宕机或网络分区时, 系统仍可对外提供服务. 但高可用性带来了高跨域复制开销, 影响系统的性能[20]. 并且共识协议要求副本间保持实时同步, 同样影响系统的性能.

由此可见, 跨域部署时性能、扩展性和可用性带来的挑战集中在区块链系统的共识算法、并发控制和分片策略上. 为了更好地理解这些挑战, 本文首先介绍联盟链的工作流程: 首先, 共识协议的主节点将交易打包生成区块, 经由**共识算法**复制到所有从节点. 在节点收到经过共识的区块后, 节点确定性地使用**并发控制**算法执行区块内交易, 更新本地账本, 所有节点的账本在应用区块前后分别保持一致. 为了提升可扩展性, 一些区块链进一步对**账本分片**, 使用提交协议协调跨分片交易.

从上述描述可以看出, 设计高性能的跨域联盟链系统需要选择合适的共识算法、并发控制和分片策略. 为了进一步明确它们的重要性, 本文接下来介绍使用传统方法在跨域区块链系统中面临的挑战:

1) 共识算法跨域通信开销: Fabric[32]、Quorum[33]等传统联盟链系统使用 PBFT[34]、Raft[35]等扁平共识协议. 首先, 扁平共识不考虑数据中心内与跨数据中心的网络拓扑异构, 跨域通信开销大, 共识效率低. 其次, 跨域区块链通常部署在横跨全国或全球的多个数据中心上, 用来服务不同地域的用户. 但扁平共识协议使用唯一的主节点对交易排序, 不具备写扩展性. 而且用户写请求需要跨域发送到主节点定序, 对于需要多次交互的交易, 数次跨域往返通信极大提升了交易的延迟. 最后, 共识协议普遍使用主节点将日志复制到所有从节点, 从节点数量较多时日志复制会成为瓶颈.

2) 并发控制性能: 由于涉及到庞大节点数量和复杂网络状态, Bitcoin[36]、Ethereum[37]等公有链的共识阶段是系统瓶颈, 而交易执行开销可忽略不计. 因此, Bitcoin 等公有链采用串行方式简单调度策略, 不需要并发方式来提高交易执行的效率. 但是, 随着流水线、批处理等优化技术的应用, 联盟链的共识效率显著提高, 串行交易执行模式逐渐成为系统瓶颈, 有必要采用更高效的并发控制技术. 现有联盟链系统使用多种并发控制算法提升吞吐率, 但是不同并发控制算法在吞吐率、提交率、延迟等方面存在显著差异. 例如, Fabric 将乐观并发控制 (OCC) 引入联盟链, 但在跨域高延迟环境下终止率较高. NeuChain[38]使用确定性并发控制保证高吞吐率, 但交易终止率较高. OrderlessChain[39]使用无协调一致性 (CRDT), 但其不支持可串行化隔离级别.

3) 分片策略的同步开销: 分片技术可以提升系统扩展性, 在跨域部署时能获得更好的性能. 通过将账本分为若干可并行提供服务的分片, 并将这些分片部署在不同集群, 系统的吞吐量可以随节点数量的上升而上升. 但处理跨分片交易需要相关分片的所有副本参与协调, 在跨域部署时会产生分片间同步开销, 降低系统的性能. 例如: RapidChain[40]和 AHL[41]采用两阶段提交 (2PC) 协议, 每个阶段都需要在分片内的所有副本间达成共识. Caper[28]和 Sharper[42]将共识协议作为提交协议, 需要在参与分片的所有副本间进行全连接通信. 除此之外, 在系统存在较多跨分片交易时, 性能甚至低于全副本部署[43]. 例如, 分片区块链通常采用 2PC 保证分片间交易的原子性. 在跨域部署时, 为了避免单个分片因为区域宕机而失效, 通常会在多个数据中心部署相同分片的多个副本. 此时分片内共识需要跨域完成. 若分片共识采用 PBFT, 则需要 1.5RTT 跨域通信. 而典型 2PC 甚至需要三次分片内共识才能完成[29, 44], 庞大的通信开销极大降低了系统的性能.

由此可见, 设计高性能的跨域联盟链系统需要解决上述三点挑战. 虽然现有公有链和联盟链已经发现并解决了部分问题[45-48], 但这些研究并未在跨域部署的环境下系统全面地讨论和总结联盟区块链面对的挑战和对应解决方法. 综上所述, 本文重点讨论跨域联盟链的相关技术, 第 1 节至第 3 节将从共识、交易执行和分片方面, 详细介绍当前区块链系统跨域部署时的挑战和解决方案; 第 4 节总结全文, 并给出今后工作展望.

1 跨域共识协议

PBFT[34]、Raft[35]、Paxos[49] 等传统共识协议在跨地域部署时性能较差，主要原因包括: 1) 这些协议没有利用跨域网络的拓扑结构, 节点间全连接通信会产生大量额外网络开销[50]. 2) 用户亲和性设计不足. 用户通常分布在跨地域的多个数据中心[22], 而仅存在一个全局主节点打包区块, 导致用户跨域发送交易增加延迟. 同时, 全局仅存在一个主节点也会形成共识瓶颈[51]. 3) 在共识时, 以防止拜占庭行为导致的数据丢失, 区块传输多份造成较大跨域通信开销[38]. 针对以上问题，现有系统提出了分层共识算法、多主共识算法和异步共识算法[52, 53]等解决方案, 并提出和共识时区块传输优化策略减少跨域网络环境对共识协议的影响.

1.1 分层共识协议

分层共识协议通过增加域内高速网络的使用减少跨域通信开销. 通过将节点按照数据中心分组并在每个组内选出一个主节点代表该组与其他组通信, 分层共识算法解决扁平全连接的共识不能很好适应跨域网络拓扑的问题. 区块首先在组内的高速网络下进行共识, 以防止节点的拜占庭行为, 随后被发往其他组完成分组间复制, 以防止某些分组由于故障等原因宕机导致区块丢失.

按照分组间区块的复制策略可以将分层共识算法分为两层共识算法和树形共识算法.

1.1.1 两层共识协议

两层共识算法针对跨域公网环境和域内高速网络环境设计, 在降低总体延迟的同时提升了吞吐率. 该方法简单直观延迟较低, 发送方分组 (通常由分组的主节点) 将区块通过共识协议或直接发送到接收方分组.

Steward[50]首次提出了两层共识协议. 该协议在组内运行拜占庭容错 (BFT) 的共识协议 PBFT, 保证分组内节点做出决策的一致性, 同时屏蔽拜占庭节点. 每个组选出一个主节点代表该组与其他组通信, 这些主节点间使用在仅崩溃容错 (CFT) 共识协议 Multi-Paxos, 确保在分组宕机时可用. 为了防止分组内主节点的拜占庭决策, 主节点产生的决策必须经由组内共识后才能发送到其他分组. 为了防止其他分组的接收者节点是拜占庭节点而拒绝接收消息, 主节点需要将共识消息广播到每个组的所有节点. 当主节点拜占庭或宕机时, 分组内其他节点启动 PBFT 的视图切换 (view-change) 来选取新的主节点替换该拜占庭节点.

Steward 的两层共识协议将跨域通信复杂度从 $O(N^2)$ 降到了 $O(M^2)$, N 和 M 分别为节点数和组数. 这是因为跨域通信开销由所有节点的全连接通信降到了分组主节点间的全连接通信. 但是, 将节点分组也相应地降低了 BFT 容错. 相对于 PBFT 等扁平共识协议最多容忍的全局 1/3 拜占庭节点, Steward 只能容忍分组内不超过 1/3 拜占庭节点和不超过 1/2 分组宕机.

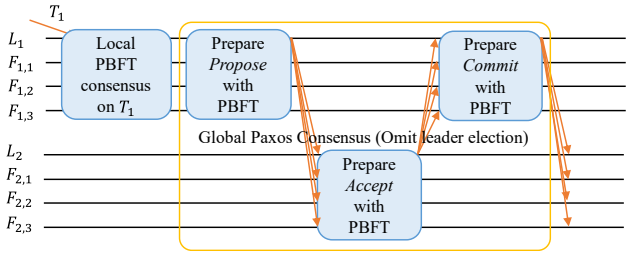

图 1 Steward 的 Paxos 复制流程

如图 1 所示, Steward 的两层共识算法流程如下: 1) 为避免跨域网络协调开销, 用户将请求发送到位于相同分组的主节点. 2) 在收到用户请求后, 主节点构建包含请求的 Propose 消息, 并发起全局 Paxos 将其复制到其他分组. 为保证请求有效性, Propose 消息需要通过本地 BFT 共识并收集到 $f+1$ 签名 (f 为分组内拜占庭节点数). 3) 当收到 Propose 消息时, 分组主节点构建 Accept 消息作为回复. 因为作为 Accept 消息输入的 Propose 消息已经被确定, 领导者仅在收到分组内 $f+1$ 的签名回复后即可将 Accept 消息广播到其他分组. 4) 当收到超过半数其他分组的 Accept 消息时, 即可认为请求已经全局完成复制, 此时该请求可以被安全地提交和执行.

Blockplane[54]设计了 BFT 容错的分层框架, 针对跨组复制和组内提交分别设计了对应 BFT 原语, 并基于上述原语提出一种跨域共识算法. 类似于 Steward, Blockplane 的两层共识算法分别采用 PBFT 和 Paxos.

GeoBFT[31]同样采用了两层共识架构, 并进一步减少了跨域通信开销和跨域协调开销. 如图 2 所示, 在减少跨域通信开销方面, GeoBFT 在跨域日志复制时, 主节点仅将日志复制到每个接收域的 $f+1$ 个节点上, 而非复制到所有节点. 由于 $f+1$ 节点至少包含一个非故障节点, 节点本地广播收到的请求可确保域内所有诚实节点收到日志. 在减少跨域协调开销方面, GeoBFT 采用了直接跨域广播日志的方法, 并没有使用全局 CFT 共识. 采用直接广播可以将跨域协调次数由 3 次降低到 1 次 (1.5RTT 降低到 0.5RTT), 但牺牲了区域容错. GeoBFT 不能在网络分区或数据中心停电等情况造成的组失效时保证共识的正常进行. 为防止发送方的领导者为拜占

庭节点, GeoBFT 设计了远程视图切换协议. 当领导者拒绝跨域转发消息, 或领导者的跨域网络故障时, 接收域可通知发送域更换领导者. GeoBFT 还采用了多主共识算法, 所有分组可以并发接收并执行区块 (见 1.2 节).

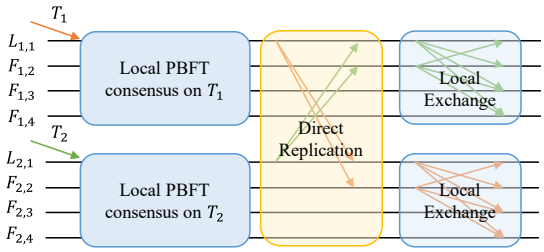

图 2 GeoBFT 共识架构

D-Paxos[55]和 C-Raft[56]为崩溃容错的两层共识算法. D-Paxos 采用 Multi-Paxos[57]做底层共识并采用 Paxos 做顶层共识. D-Paxos 允许每个分组并发收集并缓存用户交易, 这些组轮流充当全局 Paxos 的主节点. 当一个分组成为全局主节点时, 该分组将缓存的一批交易使用全局 Paxos 广播到其他分组. 该方法让用户可以在本地处理交易, 而不必跨域将交易发给全局 Paxos 的主节点. C-Raft 采用 Fast Raft[56]作为顶层和底层的共识协议, Fast Raft 优化了 Raft 的消息传输流程, 可以在没有并发提案的情况下将提交延迟由 1.5RTT 缩短到 1RTT. 在消息丢失或存在并发提案时, Fast Raft 回退到经典 Raft[35]共识.

表 1 两层共识算法和系统

系统	年份	容错模型	底层共识	顶层共识
Steward[50]	2008	BFT	PBFT	Multi-Paxos
Blockplane[54]	2019	BFT	PBFT	Paxos
GeoBFT[31]	2020	BFT	PBFT	直接广播
D-Paxos[55]	2016	CFT	Multi-Paxos	Paxos
C-Raft[56]	2020	CFT	Fast Raft	Fast Raft

1.1.2 树形共识协议

针对网络拓扑大于两层的情况, 树形共识算法可以进一步匹配网络基础设施提升共识性能. 树形共识算法中, 分组可以多次聚合形成更大的分组. 区块经过多次分组间的转发到达所有分组. 多次转发的传输方式极大减轻了发送方分组的带宽压力, 但是由于分组间需要多次跨域通信, 共识的延迟相对较高.

Canopus[58]为 CFT 共识协议, 运行在一个三层的网络拓扑结构下. 其假设节点分布在多个数据中心, 一个数据中心内的节点分布在多个机架上. 相同机架内节点间网络优于数据中心内机架间网络, 而数据中心内机架间网络优于数据中心间网络.

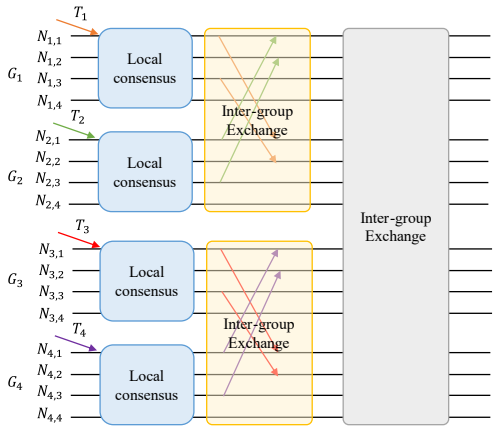

图 3 Canopus 共识架构

为适应这种多层网络拓扑, Canopus 将节点按机架分为 CFT 小组, 将这些小组作为叶子节点构建了一个仅有叶子节点真实存在的树 (Leaf-only tree, LOT), 如图 3 所示. 其中, G_1,G_2,G_3,G_4 为 4 个分组, 每个分组内包含 4 个节点, 构成了 LOT 第 1 层的四个叶子节点. LOT 的第 2 层将 G_1,G_2 分为一组, G_3,G_4 分为一组, 而第 3 层为 LOT 的根节点, 包含所有分组. 对于交易 T_1 而言, 节点 $N_{1,1}$ 收到交易后先在分组 G_1 内完成 Raft 复制. 随后, 分组 G_1 和 G_2 交换交易 T_1, 确保 T_1 在数据中心内完成复制. 最后, 数据中心间交换 T_1, 完成全局共识.

Canopus 很好适应了跨域复杂的网络拓扑. 为减少跨域通信次数和开销, Canopus 同样牺牲了区域容错, 仅在分组内使用 Raft 共识, 可以容忍节点宕机. 而分组间采用直接广播交易的方式. 在分组崩溃时, 系统需要需要等待崩溃分组恢复, 因为其他分组无法确定该分组已经崩溃还是仅由于分组间消息传输缓慢(系统运行在部分同步网络模型[59]下)导致的延迟. 总而言之, 分组间需要针对是否等待宕机分组达成一致.

RCanopus[60]为 Canopus 的拜占庭容错版本, 其将位于相同机架的节点分组, 分组内运行 Raft 共识. 而分组间首先使用 PBFT 聚合形成拜占庭分组, 以容忍拜占庭节点. 最后, RCanopus 在这些拜占庭分组上运行 LOT 传输交易. 值得注意的是, 由于分组使用的 Raft 共识并不能容忍拜占庭错误, 因此当分组内存在拜占庭节点时, 整个分组即被认为是拜占庭分组. 除此之外, 针对 Canopus 不能容忍分组崩溃, RCanopus 在每个拜占庭内选出一个主节点, 在这些主节点间运行 PBFT 以维护分组信息. 当分组崩溃时, 分组信息会被及时更新, 节点能一致地决策是否等待或应用来自崩溃分组的交易.

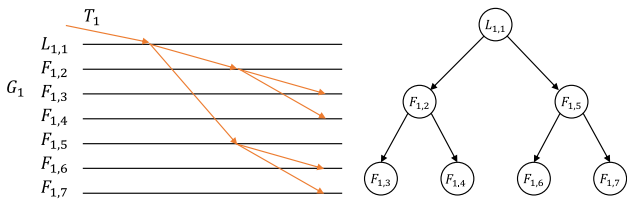

图 4 Kauri 的树形区块复制(左)和 7 节点时的节点拓扑(右)

Kauri[61]改进了 HotStuff[62]的全广播消息传输方式以适应跨域部署, 其使用平衡多层树广播交易提案并聚合回复, 在 7 节点时的通信模式和对应拓扑结构, 如图 4 所示. 对于内部节点故障 (如 $F_{1,1},F_{1,5}$), Kauri 采用了高效的重配置算法, 避免由于这些节点连续拜占庭而导致的故障. 其还使用了流水线技术 (见 1.4 节) 进一步提升性能, 允许多轮共识并发进行, 以应对沿拓扑路径多次跨域通信造成的高延迟.

Byzcoin[63]采用多层的树形通信减少 PBFT 在公有链上部署的网络开销. ByzCoin 也采用 CoSi[64]多重签名(Multi-Signatures)聚合组内节点签名, 减少 PBFT 的 Prepare 消息和 Commit 消息的签名传输开销 (见 1.4 节).

与 Byzcoin 和 Kauri 采用的固定宽度的拓扑不同, Omniledger[65]提出的 ByzCoinX 使用固定高度的拓扑结构. 其将从节点分组, 每个组选出一个节点充当分组的主节点. 消息传输由 PBFT 主节点传输到分组的主节点, 在由分组主节点转发到分组内从节点. 为减少分组主节点拜占庭的情况, ByzCoinX 结合了 RandHound[66]和可验证随机函数[67] (Verifiable Random Function, VRF)保证节点分组和选择分组主节点的随机性, 即拜占庭节点不能预知和指定其所在的分组和分组的主节点.

表 2 树形共识算法和系统

系统	年份	容错模型	底层共识	顶层结构	分组容错
Canopus[58]	2017	CFT	Raft(多主)	LOT	否
RCanopus[60]	2018	BFT	PBFT(分组级别)	LOT	是
Kauri[61]	2021	BFT	广播网络	多层树	是
ByzCoin[63]	2016	BFT	广播网络	多层树	是
ByzCoinX[65]	2018	BFT	广播网络	两层树	是

1.2 多主共识协议

对于 PBFT、HotStuff [62]等主从架构共识协议，用户需要将交易发往全局唯一的主节点排序并生成区块，唯一主节点会产生单点瓶颈. 1) 用户通常分布在不同的地理区域，写请求跨地域传输到主节点会产生额外的延迟，尤其当请求需要多次和用户交互的情况. 2) 在网络波动时，部分节点的副本并非实时最新. 因此，读请求也需要发往主节点处理，避免读陈旧数据. 但这会加重主节点负载[38, 51]，恶化单点瓶颈效应. 3) 主从共识仅存在一个主节点发送请求，在上传带宽受限时，主节点将区块发送到多个从节点同样会成为瓶颈.

多主共识协议解决了上述问题，在跨域部署时能拥有更好的性能. 首先，多主共识所有节点均为主节点，不同区域用户的写请求均能发往本地节点处理[58]，端到端延迟较低. 其次，多主共识具有读可扩展性. 由于节点副本间保证了强一致同步，对任意副本的读请求均能返回最新数据. 最后，所有节点同时发送不同的交易，可以同时利用所有节点的上传带宽，避免了主从架构的区块复制的单点瓶颈问题.

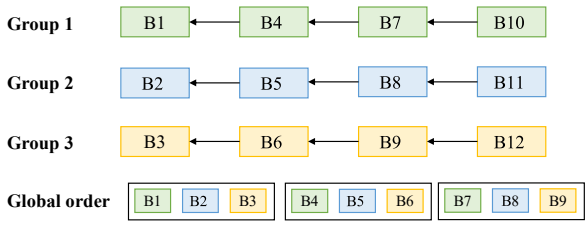

图 5 多主共识算法按轮同步确定全局顺序

以 GeoBFT[31]为代表的大多数多主共识算法使用按轮同步的排序方式协调多个节点(或分组)打包的区块. 每个分组独立收集本地用户的交易，在打包生成区块后与其他分组交换 (见 1.1.1 节). 如图 5 所示，在交换结束后，每个节点都会拥有所有分组提议的所有区块，其中每个分组都提议了一条子链. 为保证所有节点都会以相同的顺序执行这些区块，区块执行以轮的方式进行. 在一轮中，节点必须收到每条子链的下一个区块后才会按照分组 ID 的顺序执行这些区块.

NeuChain 使用绝对时间 epoch (约为 20ms) 划分区块，节点接收交易后会首先确定当前的 epoch. 在 epoch 结束后，所有节点会交换它们收到的交易. 具体来说，每个节点使用一个 Raft 实例保证广播的原子性. NeuChain 使用交易 ID 确定性地为同一轮内的交易排序，交易 ID 使用交易签名和 epoch 号的哈希生成，确保任意两个交易均有不同的 ID. 在执行时，NeuChain 采用了 Aria[68]确定性并发控制算法，节点在收到属于当前 epoch 的一批交易后即可立即执行，无需像 GeoBFT 一样需要按照节点 ID 的顺序来按序执行(见 2.3 节).

Canopus 和 RCanopus 同样采用了按轮执行的策略，分组内每个节点在一个 epoch 内均可提议一个交易，交易通过 Raft 在分组内进行原子广播，并按照 LOT 树形拓扑结构进行分组间交换. Canopus 使用 Proposal 号对同一轮内的区块排序，每个节点提议的区块都拥有随机 Proposal 号，当收到一轮内所有的区块后，可以根据 Proposal 号对日志进行确定性排序.

按轮同步简单易于实现，但是当某些节点因为拜占庭等原因发送区块的速率较慢，此时节点需要等待直到收到这些缓慢的节点的区块后才能开始执行，极大影响了系统性能. Density[69]缓解了缓慢者会影响系统吞吐率的问题. 因为按轮同步的排序规则假设了各个组复制日志的速率相同，多主共识更容易受到拜占庭节点的影响，由于要求所有副本均保证同步，任意拜占庭节点均可以通过在共识前插入一定的延迟拖慢共识[69]，增加同步开销来降低系统性能. 若拜占庭节点以较慢速率复制日志会导致该轮不能按时结束，进而阻塞排序. 因此，Density 将区块复制和排序解耦，选出一个全局的节点 (O-instance) 负责为区块排序，此时虽然节点间区块复制实例 (D-instance) 的速率不同，但由于区块的执行顺序是由该全局节点收到这些区块的顺序决定，复制缓慢的区块会被放在后面执行，而不会阻塞已经完成复制的区块.

表 3 总结了多主共识算法和系统. 多主共识协议与分层共识协议和容错机制正交，这是因为大多数多主共识仅仅使用现有主从共识协议作为模块，多个主从共识协议同时提出区块，并使用一个确定性规则确定不

同共识协议提议的区块间的全局顺序.

表 3 多主共识算法和系统

系统	年份	是否为分层共识	容错	全局排序
GeoBFT[31]	2020	两层共识	BFT	按轮同步
NeuChain[38]	2022	非分层	BFT	按轮同步
Canopus[58]	2017	树形共识	CFT	按轮同步
RCanopus[60]	2018	树形共识	BFT	按轮同步
Density[69]	2022	非分层	BFT	动态确定

1.3 异步共识协议

上述跨域共识协议均假设网络模型为同步(synchronous)或部分同步(partial synchronous)的, 使用计时假设(timing assumptions)来保证系统的安全性或活性. 1) 同步网络模型假设存在已知的有限时间界限 Δ, 正确节点发送的消息最多会被延迟 Δ 后传送到其他所有正确节点. 对于运行在同步网络的共识协议而言, 若该假设被打破, 则无法保证安全性. 因此, 这些共识协议通常运行在 Infiniband 等高速或小范围的网络下, 或使用可信执行环境 TEE 的 CFT 模型下. 2) 部分同步网络模型下的 BFT 共识协议假设网络在大部分时间保持同步, 即正确节点间的消息传递存在一个有界的延迟上限 Δ, 该上界是未知的, 并且能由敌手任意选择. 由于其最符合真实的网络环境, 绝大多数共识协议运行在部分同步网络模型下. 但是, 在假设被打破即网络异步时, 这些协议只能保证安全性而不能保证活性.

在跨地理区域甚至跨洲部署时, 节点间的通信延迟很可能不稳定, 并且差异较大[53]. 异步网络模型不对网络状态进行任何假设, 消息可以被任意的延迟. 因此, 异步共识协议能够实时跟踪当前网络的状况, 在网络设施质量较差时提供最佳的吞吐量.

但是, 保证实时最佳吞吐量的代价是异步共识协议需要较长的时间才能达成共识. 由于 FLP 不可能定理[70], 即在异步网络模型下, 即使只有一个进程失效, 也没有任何算法能保证在不失败的情况下达成一致. 因此, 异步共识协议无法确定性达成共识, 均为随机化协议(randomized agreement). 由于每次仅有一定概率达成共识, 这些共识算法[52, 53]每次共识可能进行多轮(rounds). 例如, 文献中的 ABA 共识(Asynchronous Binary Byzantine Agreement, 异步二值拜占庭共识)[71]达成共识所需要轮数的期望为 4, 而在 Dumbo2[52]的 MVBA 共识(multi-valued validated Byzantine agreement, 多值拜占庭共识)同样需要运行约 3 轮 ABA 才能达成共识.

许多异步共识协议[52, 53]均基于文献[72]的经典模型, 其中, HB-BFT[53]为第一个实用的异步 BFT 共识协议. 类似多主共识协议(1.2 节), HB-BFT 中的每个节点均能提议交易. HB-BFT 的核心为 ACS 模块(异步共同子集, Asynchronous Common Subset), 用于在每次共识时选择一组节点(至少 $n-2f$ 个)提议的交易附加到全局日志中作为输出, 如图 6 所示. ACS 模块由 n 个 RBC[73]实例(Reliable broadcast) 和 n 个 ABA 实例组成, 其中 n 为参与节点的数量. RBC 实例负责保证交易广播的原子性, 即强制拜占庭只能崩溃或向正确节点广播相同的交易[73]. 对于一个 ABA 实例, 每个正确节点均需要向该实例输入 0 (还未收到该实例对应 RBC 的结果)或 1(收到该实例对应 RBC 的结果), 而拜占庭可以崩溃或随机输入 0 或 1. 简言之, 每个 RBC 实例对应的 ABA 实例负责决策出该 RBC 是否按时完成.

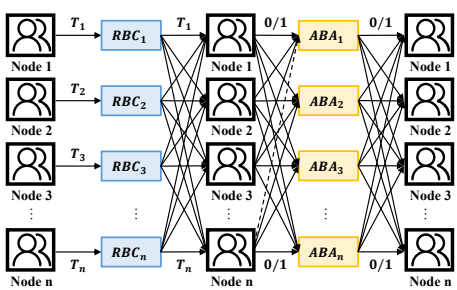

图 6 HB-BFT 的异步共同子集(ACS)结构

HB-BFT 的运行流程如下: 1) 个节点从交易池中随机选择一定数量的交易, 并使用共享公钥的门限签名[74-77]加密作为 ACS 的输入. 加密可以防止拜占庭节点得知 ACS 输入的内容, 进而有选择地共识某些交易(censorship attacks). 2) 节点 i 将其选择的交易输入到它自己的(第 i 个)RBC 实例, 广播到其他节点. 同时, 每当它从第 j 个 RBC 实例得到输出时, 它在对应的(第 j 个)ABA 实例输入 1. 当其向 $n-f$ 个 ABA 实例中输入 1 后, 其向剩余的 ABA 实例中输入 0. 这是因为剩余的 f 个 RBC 实例对应的输入节点可能为拜占庭节点, 即可能永远不会得到输出. 当第 j 个 ABA 实例完成后, 节点 i 根据输出(0 或 1)决定是否保留第 j 个节点提议的那组交易. 3) ABA 实例的共识结果唯一, RBC 保证了选择保留的交易唯一, 节点解密交易并根据节点 ID 对交易排序生成确定性的区块.

Dumbo[52]指出, HB-BFT 在共识时的主要开销为 ABA 开销. 虽然单次 ABA 的共识开销恒定, 但是每当在系统内每增加一个节点时, 系统就需要额外运行一个 ABA 实例, 对是否保留该实例的交易输入达成共识. 简言之, 虽然运行一个 ABA 共识的开销恒定, 但是因为每个节点 CPU 线程数和网络资源有限, 并发运行 n 个 ABA 实例的开销为 $O(\log n)$ [78]. 因此, Dumbo1 将需要的 ABA 实例的数量由 n 个减少为 κ 个. 简言之, 其选择 κ 个节点作为主节点, 用于决策哪些节点广播的交易包含在最终区块中, 而这 κ 个主节点仅需要 κ 个 ABA 实例即可达成共识. Dumbo2 进一步使用 MVBA 而不是 ABA 提升性能. 虽然 MVBA 的共识复杂度较大($O(n^2|m|+\lambda n^2+n^3)$, 其中 m 为消息大小, 通常大于 $\lambda n\log n$), 但是通过减少 MVBA 共识输入数据的大小, Dumbo2 的共识开销可以低于 HB-BFT.

文献[79]指出, 以上共识协议无论不论交易池内是否由交易, 都需要所有节点参加每个 epoch, 并且 ABA 协议在大规模部署时开销较大, 并为此提出了 MyTumbler, 保证了仅有少数节点提议交易时系统的快速运行.

1.4 共识时的通信优化

BFT 共识协议受限于节点间网络通信的带宽和延迟. 因此, 联盟链共识算法提出了数种方法减少节点(或分组)间的跨域通信开销, 包括硬件加速, 纠删编码, 门限签名和多重签名, gossip 和流水线技术.

1.4.1 硬件加速

在共识协议中, 最常用的硬件加速方法为使用远程内存直接访问 RDMA 技术, 用于提高网络带宽利用率, 减少节点间通信的延迟和降低 CPU 负载. 不同于传统 TCP/IP 通信协议, RDMA 依赖于 Infiniband 交换机等特殊硬件, 可以让本地 CPU 直接访问远端内存存取数据, 而不需要远端 CPU 参与. Dare[80], APUS[81]利用单边 RDMA 原语设计了强一致的共识协议. 由于 RDMA 的低延迟和不需要远程 CPU 参与消息传输, 极大提升了共识的性能. 为了保证系统的扩展性, 并能高效应对不同类型的工作负载, FlexChain[82]采用云原生的池化思想, 即利用数据中心提供的计算池, 内存池和块存储池, 重新设计了系统架构. 对于数据中心内节点通信, FlexChain 同样使用了 RDMA 技术, 减少存储和计算资源解耦后的性能损失. 除此之外, 可编程交换机也可以提高网络通信效率和减少延迟, 其允许开发者自定义数据平面的处理逻辑, 从而实现更为精细和高效的数据处理和网络流量管理. NOPaxos[83]使用可编程交换机对消息排序, 消除了 Paxos 协调者间的消息交换. 它将延迟减少到一个 RTT, 也是基于服务器解决方案的理论下限.

可编程交换机和 RDMA 等硬件加速技术在单个数据中心内能显著提升共识协议的性能, 但是, 在跨地理区域联盟链中, 上述硬件加速方法仍然面临着诸多挑战: (1) 首先, 数据中心间的网络通常难以实现硬件加速技术所依赖的特定网络硬件配置. 由于广域网 (WAN) 通常涉及多个不同的网络运营商和服务提供商, 每个运营商可能采用不同的硬件和网络技术, 这种多样性和不一致性使得统一部署特定的硬件加速技术变得极为困难. (2) 其次, 联盟区块链内的不同组织可能选择不同的云服务提供商, 这些提供商的数据中心采用的网络硬件也不一定完全相同. 因此, 可能需要适配多套硬件实现联盟区块链的硬件加速功能. (3) 最后, 对于 BFT 协议的硬件加速支持相对较少, 这限制了在跨域联盟区块链场景中部署这些技术的可能性.

1.4.2 纠删码技术

纠删码技术[84]可以降低日志复制带来的网络开销和存储开销, 被广泛应用在区块链系统中. 纠删码将日志分为 m 个大小相同的数据片段, 并额外编码出 $n-m$ 个冗余的校验片段, 其中 n 为节点总数, 副本在拥有任

意 n 中的 m 个片段即可重构出原始日志. 广泛使用的编码方式为 Reed-Solomon 纠删码[85].

CFT 共识协议中, RS-Paxos[86]是首个使用纠删码技术的协议, 用于减少日志复制的通信开销和存储开销. 其将每条日志打散成 m 个数据片段并编码出 $n-m$ 个校验片段, 每个节点仅接收和存储一个对应片段. 为了保证正确性, RS-Paxos 的每次读操作和写操作都必须在 Q_r 和 Q_w 个副本上完成后才可提交. 并且要满足 $Q_r+Q_w-n\geq m$, 即保证读取的副本集合 Q_r 至少包含 m 个最新的写入片段, 即保证恢复出最新的日志. 当 $m=1$ 时, RS-Paxos 退化为经典 Paxos 算法, 并拥有相同的 FT, 即容忍不超过半数节点宕机. 但是, 由于每个节点仅存储 1 个片段, 为保证系统在 f 个副本宕机时正常运行, 系统需要至少 $f+m$ 个正确节点 (此时若其中 f 个节点宕机并丢失数据时, 仍有 m 个片段未丢失, 进而能恢复出原始数据), 降低了系统的 CFT 容错.

为解决 RS-Paxos 容错度较低的问题, 当系统中正常节点大于 $f+m$ 时, CRaft[56]使用与 RS-Paxos 相同的编码方传输日志. 当正常节点小于 $f+m$ 时, 采用传统 Raft 的策略复制日志. ECRaft[87]和 HRaft[88]进一步改进了 Craft 的编码策略, 在任何情况下系统均使用纠删码存储. 在节点故障时, HRaft 将发往故障节点的片段分配到正常节点上, 而不是回滚到 Raft 的全复制策略. FlexRaft[89]通过在心跳或日志复制时实时侦测正常节点数量来动态改变纠删码编码模式, 保证任何情况下编码模式均为最优, 最大化减少了系统通信开销和存储开销.

BFT 共识协议也广泛采用纠删码减少日志复制开销. RapidChain[40]共识协议的主节点使用 Gossip 协议广播纠删码编码后区块, 其他节点可以借助默克尔证明重构原始区块; Poster[90]使用纠删码改进了 HotStuff[62]共识协议. Velocity[91]将纠删码和分布式哈希表 (DHT) 结合, 提升区块链的可扩展性; 文献[92]提出了延迟复制算法, 利用纠删码传输区块, 最小化发送集群和接收者间的网络开销, 但只能用于只读负载. 在异步网络模型下, 文献[93]利用纠删码减少了消息传输的复杂度, 并使用默克尔树保证消息的完整性.

在跨域联盟链场景下, 纠删码可以显著减少账本在多个数据中心间进行复制时的带宽和存储需求. 在拥有足够数量的数据片段和校验片段的情况下, 即使部分数据中心发生故障或数据丢失, 也能保证账本的完整性和可用性. 除此之外, 我们还可以在不同的数据中心部署不同数量的数据片段和校验片段, 提供了高度的灵活性. 但是, 如何在跨域联盟链中高效地使用纠删码仍然面临以下挑战: (1) 纠删码的编码和解码过程需要额外的计算资源, 这会对参与节点造成额外负担. (2) 纠删码本身不容忍拜占庭攻击, 当且仅当所有输入片段均未被篡改时才能恢复出正确的账本. 因此, 如何高效地验证纠删片段和恢复账本的正确性也存在挑战. (3) 引入纠删码势必会增加系统的复杂性. 因此, 必须合理计算并设置副本容错, 保证系统的正确性.

1.4.3 门限签名和多重签名

一个 (t,n) 门限签名[74-77]包含了 n 个参与者节点, 其中每个节点都拥有自己的公钥和共同私钥的一部分份额. 节点可以使用自己的私钥份额对消息签名, 生成合法签名的一部分. 可以聚合多个节点对同一消息的签名份额, 生成一个聚合签名. 当且仅当 n 个节点中的任意 t 个以上的签名被聚合后, 最终的签名才能通过验证. HotStuff[62], ISS[94], SBFT[95], Saguaro[96], PoE[97]等系统采用了门限签名减少签名传输开销. 类似门限签名, 多重签名[98]也用来聚合多个节点对于相同消息的签名. 多重签名中, n 个参与者节点使用自己的私钥对消息签名. 这些签名可以被聚合, 并使用统一的公钥验证, 这样签名验证开销不会随着节点数增加. ByzCoin, Kauri, AHL 等系统使用了多重签名减少了计算和签名传输复杂度.

在跨域场景中, 跨数据中心节点间存在网络延迟高和跨域网络不可靠等问题. 由于门限签名和多重签名要求多个节点之间的协调, 以聚合有效的签名份额. 节点间的高延迟会导致协调和聚合过程中的效率降低. 除此之外, 虽然采用门限签名或多重签名可以有效提升系统的扩展性, 但是由于签名本身依旧需要所有参与节点间的协调 (不可扩展), 在节点数量较多时, 签名的聚合和管理仍然是一个挑战.

1.4.4 Gossip 协议

Gossip 协议是一种去中心化的分布式通信机制, 广泛用于区块链系统[32, 37, 40]中. 消息通过在节点间随机传播保证所有节点最终都会拥有消息的副本. 跨域区块链 (例如采用 PoW 的公有链[36]) 可以利用 gossip 协议减少公网交易和区块的传输开销. 当节点传播消息时, 节点首先会选择一定数量的邻居节点广播消息, 接收节点收到该消息并检查是否超时, 并以一定的概率选择是否转发该消息, 以此循环. 超时和概率的结合可以

让 gossip 协议在效率和可靠性之间取得平衡, 如可以通过增加广播的邻居节点的数量, 以减少某些节点在广播结束后还未收到该消息的概率, 但这也会增加网络内总体的通信量.

由于 gossip 协议不依赖于中心化节点或全局稳定的路由, 在跨域区块链系统中, gossip 保证了系统的容错性和可扩展性. 除此之外, 相比全连接广播 (all-to-all broadcast) 而言, gossip 协议可以有效减少公网的传输开销. 但是, 若 gossip 广播设置的邻居节点和转发概率等参数不合理, 会导致消息丢失 (即未成功转发到所有节点) 或产生大量冗余 WAN 流量 (广播风暴), 影响系统可用性或性能.

1.4.5 流水线技术

在传统 PBFT 共识中 (如 BFT-SMART [99]实现), 下一轮共识只会在上一轮共识完成后开始. 流水线技术可以让协议的多轮共识并发进行, 下一轮共识不必等到上一轮结束, 极大提升系统性能. 几乎所有共识协议均采用了流水线技术, 例如 RCC[100], Kauri [61], GeoBFT, SBFT [95], PoE[97]等. Parallel-Raft[101]是 PolarDB 的 Multi-Raft 实现, 它提出了乱序日志复制技术 (即日志的乱序确认、提交和应用) 来提高性能; NeuChain、FastFabric[102]、BIDL[103]、Density 等系统将区块的共识与复制分离来提高性能. 共识协议仅共识区块的哈希值, 即仅对区块哈希排序, 而真正区块会异步并行地使用 ZeroMQ 等高速消息队列完成复制.

流水线技术已经被广泛应用到跨域联盟链系统中, 如 Canopus 和 FISCO-BCOS[27], 并极大提升了这些系统的性能. 但是, 如何在跨域联盟链中高效地使用流水线技术仍然面临挑战: (1) 流水线化某些流程可能需要复杂的错误处理和回滚机制. 例如, PoE[97]提出了推测执行 (speculative execution) 技术, 其将乱序处理和流水线相结合, 交易可以在达成共识前执行. 但是, PoE 需要在共识未达成时回滚已执行的交易, 这不仅增加了处理复杂性, 还可能影响系统的整体性能和响应时间. (2) 增加的并行和乱序处理可能引入新的安全漏洞. 例如, BIDL 等系统在流水线化的同时也提出了异常处理方法并详细讨论了正确性.

2 并发执行技术

联盟链普遍采用并发控制算法执行交易, 来应对共识效率提升带来的交易执行瓶颈[104]. 现阶段联盟链主要包括四类并发控制技术: 乐观并发控制(OCC)、基于依赖图的并发控制、确定性并发控制和无协调一致性 CRDT. 上述并发控制算法在跨域部署时性能会随着域间延迟、热点数据、工作负载等变化而变化, 需要根据具体需求进行选择. 本节接下来将分别介绍上述并发控制技术和它们的优缺点.

2.1 乐观并发控制

传统基于锁的并发控制, 如两阶段锁(2PL), 在交易对记录进行读写操作前需要获取锁. 这会限制对记录的并发访问, 进而影响吞吐率. 除此之外, 锁的获取与释放也会产生额外开销. 相比较而言, 乐观并发控制 (OCC) 假设交易读写冲突较少, 即大多数交易均能提交. 因此, OCC 算法在执行过程中不会对交易上锁, 读写冲突会在交易准备提交时验证.

乐观并发控制的系统代表是 Fabric, 它在执行交易时不检测读写冲突, 直到验证阶段提交时才会串行验证是否产生读旧版本(stale read)异常. Fabric 将节点分为排序节点 Orderer 和普通节点 Peer. 系统由多个组织参与, 每个节点都属于某个组织. 如图 7 所示, Fabric 的交易执行流程由执行, 排序, 验证三个阶段构成. 交易执行需要满足特定背书条件, 并需要保证执行结果(即交易的读写集)在节点间一致.

在执行阶段, 用户将交易发往满足背书条件的 Peer 执行. Peer 在执行交易时, 在交易内记录读操作的版本, 用于稍后的验证, 写操作也缓存在交易中. 交易执行并不会修改账本状态, 因此 Peer 可以并发执行数个交易. 随后, 用户比对所有 Peer 返回的执行结果是否相同, 并将结果打包发往排序节点用来生成区块.

在排序阶段, 排序服务由多个排序节点构成, 并使用 Raft 等共识协议. 排序服务只负责打包并共识区块, 而区块内交易的合法性稍后由 Peer 在验证阶段进行检查. 排序服务的主节点负责打包区块并进行 Raft 共识, 而从节点收到的交易会转发给主节点. 在共识完成后, 所有排序节点都拥有所有未执行的区块.

在验证阶段, Peer 从排序服务按序拉取区块. 在验证区块和区块内交易的合法性后, Peer 按照交易在区块内的顺序串行验证这些交易读的记录 (Key) 是否已经被其他交易修改. 若交易不存在读旧数据异常并通过

验证, Peer 将该交易标记为有效并将修改写入状态数据库, 否则标记为失效交易. 在验证完毕区块内所有的交易后, Peer 告知用户交易的执行结果.

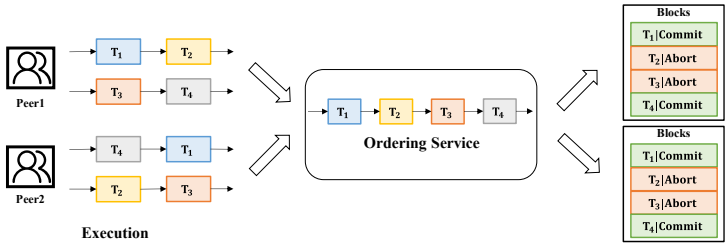

图 7 Fabric 的 OCC 执行流程

虽然 Fabric 的执行流程简单高效, 但是其存在诸多问题. 例如交易并发修改热点数据会导致高终止率. 这是因为在 OCC 执行流程中, 交易执行后会等待一段时间才能验证, 在此期间内, 交易所有读的记录不能被其他并发交易修改, 否则会产生异常需要终止, Fabric++和 Fabric#针对该问题提出了重排序技术, 即通过改变交易的验证顺序减少因为读写冲突终止的交易. Fabric++针对块内的冲突交易,在原始的排序阶段前, 用有向图的方式来表示一个块内的交易冲突情况, 通过剔除图中环上出现最多的交易来达成可串行化调度.这样不仅减少了不必要的冲突终止次数, 还提前在排序阶段而不是在验证阶段终止交易. 除此之外, 虽然交易可以并发执行, 但 Fabric 的交易验证阶段是串行进行的. 为此, FlexChain 对交易做确定性的并发验证提升性能. Fabric#和 SlimChain 进一步利用 SSI 验证交易(见 2.2 节), 避免串行提交造成的瓶颈.

2.2 基于依赖图的并发控制

依赖图 (dependency graph) 映射了不同事务间的依赖关系, 使系统能够有效地识别并发执行事务时可能出现的冲突和依赖. 基于依赖图的并发控制可分为两种, 已知事务读写集和未知事务读写集的并发控制. 如果事务的读写集已知 (即事务在构建依赖图前已经完成执行), 系统可以快速根据读写集构造区块的完整或部分的读写依赖图, 例如 Fabric#[105], 文献[104], SlimChain[106]和 XOXFabric[107]. 若构建依赖图时事务读写集未知, 可以通过推测执行[108], 预执行[109]和代码分析[110]等技术, 在获得读写集后构建依赖图, 例如 FISCO-BCOS[27]. 最常用的使用依赖图的并发控制算法为快照隔离 SI 和可串行化快照隔离 SSI[111].

快照指的是在写操作时, 不在原本的值上修改, 而是创建一个新的版本. 读操作会读取最近提交成功的版本数据. 快照隔离是并发控制的另一种方法, 它利用了每个数据项的多个版本, 每个事务所看到的数据库状态是在他开始之前提交的所有事务产生的. 快照隔离 SI 可以解决可重复读级别的问题, 但因为存在写倾斜[112]问题, 其达不到可串行化隔离级别. 写倾斜指在并发控制过程中, 多个事务对资源进行更新操作, 由于隔离性导致更新操作丢失.

可串行化快照隔离 SSI 是在快照隔离 SI 基础上支持可串行化. SSI 在数据库的版本快照上并发执行多个事务, 通过依赖图检测的方法保证事务的可串行化.由于区块链以批处理的形式并发更新账本, SSI 天然适合用在区块链系统上. 依赖图中的事务存在 RW, WR, WW 三种关系[112]. RW-反依赖为交易 T_1 先更新记录 A, 随后交易 T_2 读了记录 A 的旧版本, 此时等价执行顺序为 T_2 先于 T_1 执行; WR 依赖为 T_1 先更新了记录 A, 随后 T_2 读了记录 A 的版本, 此时等价执行顺序为 T_1 先于 T_2 执行; WW 依赖为 T_1 先更新了记录 A, 随后 T_2 在 T_1 的基础上更新记录 A, 此时等价执行顺序为 T_1 先于 T_2 执行. 因此, 若依赖图中存在环, 则无法保证可串行化.

Fabric#[105]将 SSI 引入到区块链用于在排序阶段检测交易的读写依赖, 进行重排序以提交更多的交易. 其将 SSI 定义的三种依赖关系与两交易的并发关系结合, 定义了六种读写依赖关系, 其中 n-ww, n-wr, n-rw 三个依赖关系存在于非并发交易间, c-ww, c-rw, anti-rw 存在于并发交易间. 在这些依赖中, 只有存在 anti-rw 依赖的交易会产生异常, 例如: 交易 T_1 进行对记录 A 的写操作的同时交易 T_2 开始对 A 的读操作. 后来由于交易 T_1

先于交易 T_2 提交，导致 T_2 读到了 A 的旧版本(stale read).

在 Fabric#的排序过程中，排序服务主节点可以通过分析读写依赖图判断交易是否能被提交. 与 Fabric++类似，检测图中依赖成环和冲突而不能提交的交易，并通过剔除不能提交的交易并通过重排序提高交易提交率. 由于排序阶段已经剔除所有冲突交易，Fabric#的验证阶段不需要串行进行，节点在收到区块后可以直接并发提交交易，更新状态数据库. 但是，Fabric++和 Fabric#需要为区块内交易构建依赖图. 由于构建图的复杂度高，当区块内交易数量增多时，构建依赖图所需的处理时间也随之增加，造成系统性能的降低. 除此之外，由于 Fabric++和 Fabric#均继承了 Fabric 的执行-排序-验证架构，它们的终止率存在热点的工作负载下也较高.

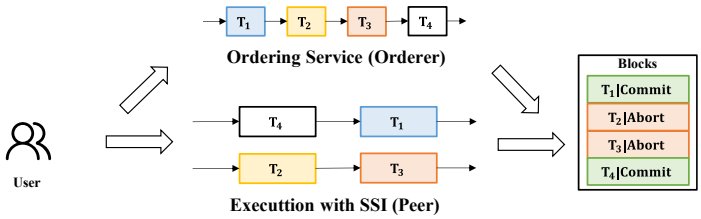

图 8 使用 SSI 并行进行排序和执行阶段

如图 8 所示，文献[104]提出一种将排序与执行阶段并发进行的架构，用户将交易发往 Peer 做预执行的同时将交易转发给其他节点和排序服务，排序服务在执行交易的同时排序交易，共识生成区块. 在区块生成后，Peer 按照区块内交易的顺序运行 SSI 并发控制，为了保证并发执行排序过程中所有节点以相同的状态提交，使用了一种基于区块高度的 SSI 方法，这种方法根据当前块的高度和操作来决定是否终止交易，以此保证可串行化，并得到确定性的执行结果.

FISCO-BCOS[27]在事务执行前确定性地构建依赖图. 为了增加事务执行并发度，FISCO-BCOS 将事务按照其调用合约划分为多个分区 (shard). 和分片区块链不同，FISCO-BCOS 的每个节点执行全部事务并保存完整账本状态，这样可以避免执行跨分片事务造成的跨域网络协调开销. 为保证不同节点上跨分区事务执行的确定性，FISCO-BCOS 使用屏障 (round) 来同步节点内这些分区的执行. 在每一轮中，每个分区使用其执行器分析分配给该分片的事务，根据事务之间的依赖关系构建有向无环图 (DAG) 用于并发执行. 由于执行器只有当前 round 该分区内事务的 DAG，当其他分区的事务调用该分区的智能合约时，事务会挂起直到下个 round 开始，确保执行器生成包含这个事务的 DAG. FISCO-BCOS 还使用了检查点来验证执行结果的一致性，若无法在一定时间内完成验证，该区块会被重新执行.

2.3 确定性并发控制

在执行同样一组交易时，由于交易对锁的竞争、操作系统对线程的调度，网络延迟等原因，非确定性并发控制无法保证这组交易的每次执行的结果都一致. 上述采用非确定性并发控制的区块链系统中，主节点必须在执行交易后共识区块，验证者按照执行顺序验证区块. 非确定性并发控制缺点如下：共识前需要先执行交易，确认延迟较高；按照给定顺序验证区块限制了验证者的并发度；除了包含原始交易外，区块还必须包含执行结果用于验证，共识时传输执行结果会产生额外开销.

在分布式数据库中，Calvin 最早提出了基于有序锁[46]的悲观协议. 其存在一个全局的锁管理和多个 scheduler，其中 scheduler 通过顺序扫描经过共识的日志为交易获取锁，交易当获取所有锁后才能执行. 但协议的锁管理必须已知交易的读写集，读写集可通过预执行等方式获取 (与 2.2 节为读写集未知的事务构建依赖图类似)；Sparkle[113]采用试探性执行的乐观并发控制协议，在提交阶段判断交易是否满足确定性和可串行化；PWV[114]通过让交易的更新尽可能早地被其他交易可见来减少 stale read 的发生，进而提高成功率. Aira[68]利用依赖图[107, 114]实现确定性并发控制，并提出了一种确定性重新排序机制，进一步减少交易终止率.

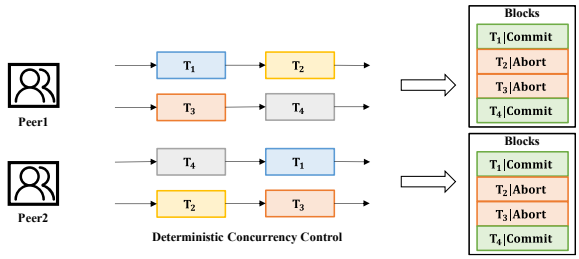

图 9 使用确定性并发控制的执行流程

图 9 展示了确定性并发控制的流程. 在收集到一个区块内的所有交易后, 全节点 Peer1 和 Peer2 并发执行这组交易而无需进一步协调. 虽然由于操作系统调度等原因, 这些节点执行交易的顺序不一定完全相同(例如, 当使用 Aria 并发控制时, Peer1 先执行 T_1 后执行 T_2, Peer2 则并发执行这两个交易, 而无需考虑先后顺序), 但是当执行完成后, 每个节点的状态都从相同的初始状态确定性地转换为相同的最终状态.

确定性并发控制的代表系统为 NeuChain[38], HarmonyDB[115]和 SChain[116]. 这些系统先对未执行的交易做全局排序, 随后复制到节点执行. NeuChain[38]在区块链场景下利用 Aria 确定性并发控制方法, 消除了排序服务导致的单点瓶颈, 极大提升了系统性能. 由于执行结果是确定性的, 与 Fabric 不同, NeuChain 的交易执行在共识后进行, 因此, 共识协议的日志复制不必包含交易执行结果, 减少了大量跨域传输开销. 但是, 由于其采用的确定性并发控制算法限制每条记录在一个区块内只能被更新一次, 在存在热点的工作负载下, 热点记录会被频繁更新, 若区块内包含多个更新该热点记录的交易 (它们间存在写-写冲突), 仅有一个能够提交. 因此, 区块内交易数量的提升会导致更多的事务间存在写-写冲突而被终止, 影响系统性能.

HarmonyDB[115]提出了基于磁盘的确定性并发控制算法 Harmony, 可以应对热点数据并能保证区块间并发. 实验表明 Harmony 不但降低了交易终止率还提升了系统的并发度.

SChain[116, 117]的一个逻辑节点由多个云服务器节点构成, 这些服务器节点分别用作排序(ordering) 、协调器(coordinator)、执行(executor)、存储(storer). SChain 采用云原生的存算分离思想, 每个逻辑节点内的执行节点和存储节点可以独立扩展. 这样, 系统吞吐量随执行节点的数量的增加而增加, 在扩展执行节点时不会涉及到存储数据的迁移. 在执行时, 用户将交易提交给排序服务. 排序服务由多个排序节点构成, 每个排序节点都能接收交易. 每个逻辑节点包含一个协调器, 负责接收排序节点共识完成的区块. 协调器使用基于有序锁的确定性并发控制, 将不相交的事务集交给数个执行节点做确定性并发执行. 为了弱化读写依赖事务对并发执行的影响, 协调器使用 Metis 图划分算法[118]来减少读写依赖事务的比例. 存在读写依赖的事务, 执行节点会在执行期间交换写入结果, 并将更新提交到存储节点中完成持久化.

SChain 采用了区块间的流水线执行, 当第一个区块执行完与第二个区块相关的交易后, 后者即可开始执行, 而不用等到前者完全完成执行, 进一步提高了并行度. 但在 Calvin 中, 由于单个协调器 (调度器) 需要给众多执行器提供完成加锁的交易, 当执行器消费交易的总效率大于单一协调器的生产效率时, 协调器本身可能会成为瓶颈[68]. 因此, 由于 SChain 并未针对 Calvin 的协调器做优化, 随着吞吐率增加或区块内交易数量的增加, SChain 的协调器可能会成为瓶颈.

2.4 无协调一致性技术

上述并发控制算法均保证了可串行化的隔离级别, 并需要系统运行强一致的共识协议保证副本一致性, 存在以下两点缺点: 首先, 很多应用并不需要可串行化的强隔离级别即可保证事务一致性[119], 即区块链可以对这类事务使用较弱隔离级别. 其次, OCC 对交易提交延迟敏感, 跨域共识导致的高延迟会显著降低其吞吐率. 而确定性并发控制和 SSI 对区块大小敏感, 区块内交易数量较多会显著降低交易提交成功率.

CRDT[120]是一种可以满足在节点间复制时, 节点间不需要额外协调即可让账本保证一致的数据结构, 通过削弱副本间数据的一致性(副本间仅为最终一致)、避免副本间协调提升性能, 同时可以实现多种弱隔离级

别. 由于满足 CRDT 性质的交易均能提交, CRDT 交易提交的成功率不受交易延迟和区块大小的影响. 目前 CRDT 可以应用到选举投票, 拍卖等很多不需要协调而保持一致性的场景, 并可以实现读已提交(Read committed), 读未提交(Read uncommitted)等多种隔离级别[25].

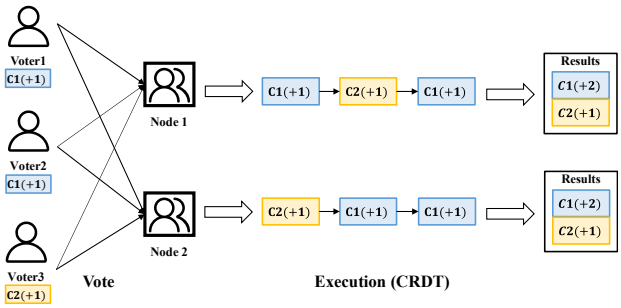

图 10 CRDT 并发控制的执行流程

图 10 展示了一个 CRDT 的投票应用部署在节点 n_1 和 n_2 上, 其中 n_1 和 n_2 均记录了完整账本. 存在候选人 c_1, c_2 和三名投票者 v_1, v_2 和 v_3. 其中, v_1 和 v_2 投给 c_1, v_3 投给了 c_2. 虽然节点 n_1 和 n_2 分别以不同的顺序收到了三个投票请求(n_1 按序收到 v_1, v_2 和 v_3, 但是 n_2 逆序收到 v_3, v_2 和 v_1), 由于投票具有累加性质, 两个节点的最终结果相同, 可以做出相同的决策.

文献[120]对数据复制过程进行建模, 首次提出了基于状态的 CRDT (CvRDT) 和基于操作的 CRDT (CmRDT) 两种等效的建模方法. CvRDT 交换完整的状态, 将本地状态与接收状态合并. 而 CmRDT 只通过向其他节点传输更新操作来传播状态. CRDT 有无协调和最终一致性的特点, 天然适用于高并发, 但单纯利用 CRDT 理论还不足以达到高性能和扩展性.

联盟链 FabricCRDT[121]将 Fabric 与 CRDT 结合, 同时支持 CRDT 交易与普通交易. 普通交易和 CRDT 交易位于相同区块, 采用不同的并发控制算法分别执行. 因此, 系统可以单独并发处理 CRDT 交易, 提升性能. 但是, 所有 CRDT 交易均经过共识协议共识, 不能发挥 CRDT 无协调一致的优点.

OrderlessChain[39]利用应用层面的约束收敛(Invariant-Confluence) 性质, 使用基于客户端的两阶段提交协议 (2PC), 用无协调一致性替代了共识阶段. 不同于 FabricCRDT, OrderlessChain 充分发挥 CRDT 无协调一致的优点, 但系统仅支持弱隔离级别, 并且节点各自保存的区块链的物理结构也因为缺乏全局共识而不能保持一致 (但账本满足最终一致).

2.5 并发执行总结

表 4 综述了上述系统所采用的并发执行技术. 由于跨域联盟链系统通常面临数据中心间节点高延迟和低带宽的挑战, 因此, 选用的并发执行算法必须充分考虑这些基本情况. 基于此, 本文建议以下并发控制策略: (1) 跨域联盟链应避免使用传统的 2PL 并发控制, 这是因为 2PL 需要在事务提交前持有锁. 在跨域部署时, 节点间的高通信延迟会显著增加锁的持有时间, 降低并发度. (2) 由于缩短了加锁时间, OCC 可以显著提升性能. 然而, OCC 仅适用于冲突率较低或处理能力较低的环境. 在需要高吞吐率且冲突率高的场景下, 由于交易提交延迟较高, OCC 会导致大量的读陈旧数据 (stale read), 进而频繁终止事务. 除此之外, 由于交易在共识前执行, 共识需要额外传输交易的读写集, OCC 在跨域网络带宽较低的场景下性能较差. (3) 基于依赖图的冲突检测相比乐观并发控制减少了事务的冲突率, 但是部分系统 (如 Fabric++) 构建依赖图时开销较高. 因此, 若跨域联盟链系统已经采用 OCC 并发控制, 结合依赖图可以有效降低终止率. (4) 跨域联盟链使用确定性并发控制性能较好. 首先, 由于执行结果是确定性的, 确定性并发控制通常在共识后执行区块内交易. 因此, 共识不需要额外传输交易的读写集, 节省公网带宽, 也不存在 OCC 高终止率的情况. 其次, 确定性并发控制不需要节点间进行额外协调来决定执行顺序, 也不存在 2PL 长时间持有锁的情况. (5) 不同于以上强隔离级别算法,

CRDT 的区块不需要使用共识协议, 不会产生高额的协调开销. 和确定性并发控制相似, CRDT 也不需要传输读写集, 执行时也不需要节点间协调, 在所有并发控制算法中性能最好. 但是, CRDT 不支持可串行化的隔离级别, 使用情形受限, 各节点按照任意顺序执行 CRDT 交易均会得到一致的结果.

表 4 并发执行技术总结

区块链系统	年份	并发执行技术	特征
Fabric [32]	2018	乐观并发控制 (OCC)	多个节点上并发执行事务, 热点数据访问终止率高
Fabric++ [122]	2019	OCC+依赖图	利用有向图表示事务执行, 剔除环中度最多的事务
Fabric# [105]	2020	OCC+依赖图	检测事务读写依赖, 用重排序提交更多事务
SlimChain [106]	2021	OCC+依赖图	存储链下进行, 使用无状态 Peer 执行和验证事务
FlexChain [82]	2022	依赖图	存算分离, 节点可以弹性扩展
文献[38]	2018	依赖图	排序与执行阶段并发进行, 使用 SSI 保证相同顺序
NeuChain[38]	2022	确定性并发控制	基于 Aria 的确定性并发控制算法, 访问热点数据差
HarmonyDB[115]	2023		基于磁盘的确定性并发控制, 可以应对热点数据
SChain[116]	2023		基于有序锁的确定性并发控制, 多个服务节点处理事务
FabricCRDT[121]	2019	无协调一致性 (CRDT)	支持 CRDT 事务和普通事务, 但仍需共识 CRDT 事务
OrderlessChain [39]	2022		节点间不需要共识, 但仅能保证最终一致性

3 区块链分片技术

分片技术旨在提高区块链的吞吐率和扩展性, 通过将账本分为多个不相交的分片, 在每个分片内独立运行共识协议. 分片区块链分摊了全副本的网络、计算和存储, 通过将账本划分为数个不相交的分片, 提升了区块链系统的可扩展性. 网络方面, 同一分片内的节点被划分到同一共识组, 不同分片的共识并发进行; 计算方面, 分片间交易并发执行, 系统吞吐率随分片数增加而增加; 存储方面, 分片内节点仅需保存分片对应的那部分账本, 大幅度减少了存储开销.

分片技术被广泛应用在跨域联盟链和公有链系统中. 在跨域联盟链中, 如 GriDB [43], Saguaro[96], Prophet [123], Sharper[42], CoChain [124], TxAllo [125]和 Ziziphus[20], 账本数据被划分为多个分片, 每个数据中心内的节点仅保存账本的一个或数个分片. 为了最小化事务的端到端延迟和提升数据亲和度, 这些跨域联盟链通常根据用户的地理未知针对性地部署分片. 与跨域联盟链的网络设施相似, 公有链的节点通常也分布在范围较广的地理区域, 节点间通信延迟较高. 因此, Elastico[126], OmniLedger[65], Monoxide[127], RapidChain[40]等公有链系统也采用了分片技术提升系统的性能和扩展性.

分片区块链中, 交易被分为分片内事务 (intra-shard transaction) 和跨分片事务 (cross-shard transaction). 由于分片内事务不涉及到其他分片的操作, 其共识和执行流程与第 1 节和第 2 节相同, 这里不做赘述. 对于跨分片事务: (1) 共识方面, 由于分片内所有节点需要对该事务的执行结果达成共识, 因此分片内共识通常采用第 1 节的共识算法, 例如 OmniLedger [65]分片内采用了树形共识, RapidChain [40]分片内采用同步网络模型下的扁平共识. (2) 执行跨分片事务要保证高效和提交的原子性. 高效是使用第 2 节的并发控制算法实现的, 例如 ByShard[44]和 AHL[41]采用的两阶段锁 2PL 和 Prophet [123]采用的确定性并发控制. 原子性即确保事务要么在所有相关分片上同时成功提交, 要么在所有分片上失败, 3.1 节将介绍以两阶段提交 (2PC) 为代表的不同种类的跨分片提交协议. (3) 存储方面, 现有分片区块链 (如 AHL[41], Elastico[126], OmniLedger[65]等) 普遍为每个账本分片对应一条子链, 分片内的节点共识负责生成该子链. 相比较而言, IOTA [128], Caper [28], Hashgraph [129], Jointgraph [130]等区块链系统的账本由有向无环图 DAG (Directed Acyclic Graph) 构成. 3.2 节将分别介绍使用多条子链和 DAG 对应的账本结构. (4) 分片数据的划分和放置的地理位置也会影响系统的性能. 例如, TxAllo 通过将频繁联合访问的事务放置在相同分片来减少跨分片事务的比例, 而 Ziziphus 将分片复制到数个地理位置相近的数据中心, 通过降低可用性来保证性能. 3.3 节和 3.4 节将详细介绍上述系统.

3.1 跨分片提交协议

现有区块链系统主要依赖于两阶段提交协议(2PC)保证跨分片的原子性. 在 3.1.1 节将介绍不同类型 2PC 的运行流程与对应系统, 并分析 2PC 在跨域部署时产生的网络开销. 为减少网络开销, 3.1.2 节将介绍其他跨

分片提交协议, 例如 Meepo [131]按 epoch 同步的策略执行跨分片交易, Monoxide [127]的原子提交协议, Pyramid [132]分层架构下的提交协议, BrokerChain [133]利用 broker 账户执行跨分片交易等提交协议.

3.1.1 两阶段提交协议

GeoChain [29], ByShard[44], AHL[41], RapidChain[40], Omniledger[65]等系统使用两阶段提交协议保证跨分片交易的原子性. 参考 ByShard[44]和 Qanaat [134]对 2PC 协议的分类, 本文将 2PC 协议分为中心化 2PC 和分布式 2PC. 其中, 中心化 2PC 存在一个中心化实体(可以为客户端[65], 某个分片[40], 或独立的协调分片[41])负责分片间的协调工作. 相比较而言, 在分布式 2PC 中, 分片间直接交换执行结果, 而无需中心化实体判断是否提交. 换言之, 集中式 2PC 需要分片将决策发往中心化实体判断是否能提交, 需要更多网络往返. 而分布式 2PC 需要更少的网络往返即可达成共识, 但由于分片间以广播的方式传输决策, 其通信开销较高.

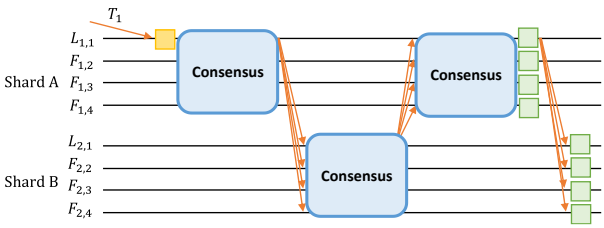

图 11 中心化两阶段提交协议

中心化 2PC 的典型系统为 AHL[41], RapidChain[40], Omniledger[65]. 如图 11 所示, 中心化 2PC 在执行跨分片交易时, 用户将交易 T_1 发往协调者Shard A. 当按照分片策略将 T_1 划分为子交易后, 将包含相应子交易的 Prepare 消息发送到对应分片Shard A和Shard B. 值得注意的是, 为避免协调者的拜占庭行为, 需要使用分片内共识或超时机制(协调者为用户, 如 OmniLedger)来消除节点的拜占庭行为.

当分片Shard B收到对应的 Prepare 消息时, 分片的主节点判断对应子交易是否能在当前分片完成提交, 并返回同意或拒绝投票到协调者. 为避免拜占庭节点左右决策, 分片Shard B内同样需要共识决策.

当且仅当所有分片的回复均为同意时, 协调者才能发起提交流程. 和 Prepare 类似, 协调者生成 Commit 消息并发送到所有参与分片, 参与分片在收到 Commit 消息后提交分片对应的子交易并更新分片负责的账本. 反之, 若某些分片(如Shard B)无法执行交易并回复拒绝时, 协调者广播 Abort 请求到所有分片以终止该交易.

AHL[41]使用可信执行环境(TEE)进行分片内共识, 减少了分片内共识的通信开销, 并提高了分片内容错. 其通过在 TEE 中维护共识消息日志使得恶意节点无法向不同节点发送拜占庭消息, 从而让拜占庭容错模型降为仅崩溃容错模型. 也就是说, 分片内只需要有 $2f+1$ 的节点数就可以保证系统的活性与安全性.

OmniLedger [65]提出了使用客户端驱动的 2PC 协议 AtomiX, 将分片间的协调转化为分片和客户端间的协调. 由于需要客户端实时监控各分片的执行结果, OmniLedger 需要高性能的客户端来驱动分片间的信息交换, 且客户端必须向整个网络广播, 通信开销较大.

ChainSpace [135]是首个支持智能合约的分片区块链,同时保护了用户的隐私性, 对 2PC 进行了改进提出了 S-BAC. 类似于 OmniLedger 的 AtomiX 协议, 但不是简单的客户端驱动, 必须有 BFT 共识过程.

RapidChain [40]利用了 UTXO 交易的性质, 将跨分片交易拆分为数个分片内的子交易, 交易可通过分片间的路由协议被送往指定分片, 减少网络通信开销. 同时, RapidChain 在分片内采用同步网络模型, 其采用的同步 BFT 共识可达到 2f+1 拜占庭容错. 其还采用流水线技术进一步提升了共识的吞吐率.

GeoChain [29]提出了一种改进的客户端驱动 2PC 协议 (inputs to output cross-shard commit IOCC) 处理跨分片 UTXO 交易. GeoChain 中的客户端首先向每个输分片发送转账交易, 输入分片共识后得到对应的交易证明并返回给客户端, 客户端可以在任何时间将附带证明的原始输入信息交易发给目标分片. 本质上是将 UTXO 交易从输入分片移动到输入分片. 其中客户端仅仅用来组合和发起交易, 如果客户端发生崩溃, 与 Atomix 锁定输入相比, GeoChain 中分片的证明仍然是有效的, 可在客户端恢复后继续使用. 如果某个输入分

片发生终止, 则终止跨分片交易, 但客户端仍保存成功的输入分片证明, 并可以在需要时发送给目标分片.

公有链系统采用中心化 2PC 会导致高额的跨域通信开销. 协调者在发起 Prepare 和 Commit 时需要进行分片内共识. 同时, 参与分片回复 Prepare 消息时也需要进行分片内共识. 由于公有链中, 每个分片内节点跨地域分布, 3 次分片内共识每次会导致 3 次分片内跨地域通信(以 PBFT 为例), 而 2PC 本身还需要 3 次跨分片跨地域通信. 因此, 在最差情况下, 中心化 2PC 会导致 12 次跨地域通信, 显著降低系统性能.

因此, 可以将分片部署在数个数据中心(而不是全部数据中心), 降低跨数据中心通信开销[20], 但这会牺牲分片的可用性. 因此, 在分片副本分布范围较广时, 分布式 2PC 通过减少了跨地域通信次数提升性能.

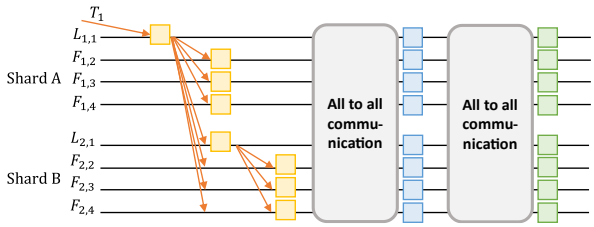

图 12 分布式两阶段提交协议

分布式 2PC 的典型代表系统为 Caper [28], SharPer [42], Qanaat [134]. 如图 12 所示, 分片 A 的主节点在收到交易后将交易广播到所有相关分片, 分片 B 的主节点在上一轮共识完成后在分片内广播交易开始共识. 随后, 分片 A 和 B 的节点进行两次全连接广播通信完成对跨分片交易的提交.

Caper[28]为针对跨应用协作的区块链系统, 由于每个应用仅维护涉及它自己的交易, 并且存在跨多个应用的分布式事务, 因此本文将其分类为分片区块链. Caper 提出了三种跨应用的交易处理办法. 1) 采用分布式 2PC 扁平共识协议, 如图 12 所示. 2) 将扁平共识替换为两层共识, 底层即可以根据安全模型采用 BFT 或 CFT 的共识协议, 上层采用类似 PBFT 的三次分组间通信达成共识. 3) 使用一组中心化的 Orderers 对所有跨分片交易做中心化排序.

SharPer [42]针对 CFT 和 BFT 模型提出了对应的跨分片共识协议: 1) 在 CFT 模型下, 当主节点收到交易请求后会向涉及交易分片的所有节点广播 Propose 消息, 分片的其他节点接收并验证消息, 如果当前节点正在等待另一个跨分片交易的 Commit 的消息, 节点将当前消息存入缓存, 否则节点会向主节点发送 Accept 消息. 由于只存在崩溃故障情况, 因此当主节点收到每个分片的 f+1 个消息并验证后, 将所有涉及交易的分片广播带有签名的 Commit 消息. 2) 在 BFT 模型下, Propose 与 CFT 环境下类似, 由于存在主节点潜在的恶意行为, 所以 Accept 和 Commit 过程需要涉及交易分片的所有节点间相互通信 Accept 消息和 Commit 消息. 如图 12 所示, 每个节点需要收到 2f+1 个消息并验证后才能提交交易.

Qanaat[134]针对多企业多分片的情况, 在跨分片交易情景下[134]同时支持企业内 (跨分片) 和跨企业 (同分片内不同副本) 的中心化 2PC 和分布式 2PC 协议. 例如, 当只有企业 e_1 拥有分片 A 和 B 的副本时, 跨分片 A 和 B 的交易可以按照图 11 的中心化 2PC 或的分布式 2PC 执行; 但是当企业 e_1 和 e_2 均拥有分片 A 和 B 的副本时, 对于中心化 2PC 协议, 同一分片内仅有一个副本 (例如企业 e_1 分片 A) 负责决策, 其他副本 (例如企业 e_2 分片 A) 仅负责接收并执行决策后的消息; 分布式 2PC 协议则将企业 e_1 和 e_2 的同分片内所有副本 (例如企业 e_1 和企业 e_2 分片 A 的所有副本) 看作一组, 由其中一个分片内的 leader 节点负责协调.

3.1.2 其他跨分片提交协议

当分片部署在不同的数据中心上时, 2PC 等提交协议会产生数次跨域通信, 影响系统性能 (见 3.1.1 节). 为此, Monoxide[127]提出了原子提交协议, 将跨分片事务划分为数个分片内的子事务, 并由发送方将这些子事务转发到目标分片执行. 接收方在收到子事务后即可立即执行, 而不必和发送方做进一步协调, 将数次跨域跨分片的通信减少为一次, 极大降低了事务的执行时间. 但是, 原子提交协议是最终一致而非强一致的, 不适用于复杂的智能合约事务[47]. 换言之, 其仅支持所有接收方的子事务均一定能提交的跨分片事务, 例如从发

送方发起的转账事务. 此外, Monoxide 还提出了诸葛连弩挖矿 (Chu-ko-nu Ming), 即允许一个节点同时参加多个分片的共识协议, 来提升分片抵抗拜占庭节点的能力. 然而, 由于共识协议本身是不可扩展的, 诸葛连弩挖矿通过增加分片内的参与节点数, 从而也会增加了区块共识完成的时间.

Meepo[131]使用 Cross-epoch 和 Cross-call 来处理复杂的跨分片智能合约事务, 事务无论涉及多少个分片均可在一轮共识中提交. 其中, Cross-epoch 代表执行一个区块需要的跨分片通信次数, 而 Cross-call 则代表在每个 Cross-epoch 期间, 发起分片会将跨分片的数据包 (包括分片 ID、智能合约地址、参数等信息) 发送给其他分片. 在最后一次 Cross-epoch 结束后, 分片收集全部的数据包执行并提交交易. 在跨域部署时, 由于分片内节点会分散在多个数据中心, 每次 Cross-epoch 都会产生一次跨域往返通信. 因此, 对于复杂智能合约, 使用 Cross-epoch 和 Cross-call 可能会带来比 2PC 更高的跨域协调开销. 除此之外, 若区块内某个事务在执行时被终止, Meepo 需要回滚整个区块重做所有事务, 这也会产生额外的跨域通信, 影响系统性能.

在跨域环境下, 跨分片提交协议(如 2PC)需要多次的跨域协调工作. Pyramid[132]针对跨分片事务进行分层设计, 由在多个分片内重叠的 b-Shard 来对所涉及的跨分片交易进行共识, 由于 b-Shard 存有所有涉及分片的账本信息, 因此可以本地对事务有效性进行验证, 对于非法事务提前判断不进行两阶段提交从而减少了跨域通信次数. 不同于每个节点只属于一个分片的传统结构, Pyramid 额外选择一些节点充当 b-Shard 同时保存 A 和 B 两个分片的交易. 在跨域异构的网络环境下, Pyramid 的分层结构可以在设计节点分配算法时将计算、存储, 通信能力较高的节点分配至分层分片系统的较高层, 以更高效的完成分层共识, 提高跨域共识效率.

CoChain [124] 允许一些分片被拜占庭节点攻占. 通过建立分片间共识组(consensus on consensus), 保证当每个分片内拜占庭节点数量少于 2/3 且共识组内失效的分组数量少于 1/3 时, 提供安全性和活性. 因此, 与传统分片区块链相比, CoChain 可以提供更高的安全性. 详细来说, CoChain 共识组内共识分为 Pre-prepare, Prepare, Commit 三个共识阶段, 在 Pre-prepare 阶段分片接收来自同一共识组受监控分片的共识结果，就结果是否有效达成分片内共识；在 Prepare 阶段使用正确分片替换拜占庭分片与每一轮共识间重新分配节点所属分片等手段保证共识协议的正确性. 在跨域部署时, CoChain 相比传统系统因为分片数量的提升拥有更高的吞吐率. 但由于共识组内节点的规模更大, 并且跨多个数据中心, 共识组内共识的网络开销大于分片内的开销, 因此 CoChain 交易提交的延迟也会相应提高[124].

Prophet[123]在分片架构下提出了一种确定性并发控制算法 (见 2.3 节) 来减少传统 2PC/OCC+2PC 的性能损失. 在跨域部署时, Prophet 通过在执行前对事务进行确定性排序来解决事务间的冲突, 避免了使用传统 2PC 执行跨分片事务, 并减少了跨分片事务在执行时所需的跨域协调和通信开销. Prophet 由各个分片中的节点自由组合产生侦查联盟,并通过激励机制使得诚实节点更倾向于留在诚实的侦查联盟中, 以减少失败概率; 由各个分片中节点组成的拥有完整合约状态的侦查联盟来进行预执行, 由序列分片根据预执行结果对事务进行确定性排序. 执行阶段在片内根据排序结果进行确定性执行, 并验证读写集与预执行结果是否一致, 如不一致则将该事务标记为无效事务, 随后在校正阶段分片间共享验证结果以提交或中断事务, 并将无效事务的节点标记为恶意节点, 以供侦查联盟中其他节点参考. Prophet 也采用了文献[104]的技术, 排序阶段可以与执行并发, 在排序时使用更加细粒度的排序, 并对读写依赖的交易进行重排序.

3.2 账本数据结构

分片区块链的账本结构可以被分为两类, 一种为多子链的存储结构, 另一种为有向无环图 (DAG) 的存储结构. 多子链的存储结构被广泛应用在大多数分片区块链中, 如 Elastico、Monoxide、Saguaro 等. 这些系统中, 每个分片维护一条子链, 子链内的区块使用哈希指针相连接, 而子链间没有做显式的连接 (如哈希指针等). 对于跨分片交易而言, 这些系统通常由发起分片保存原始交易 (如 Monoxide), 而其他分片保存涉及该分片的子交易 (如 relay transaction). 由于子链保存的内容不相交, 存储开销较低. 但是, 这也会导致跨分片交易验证开销较高. 例如, 在分片 B 验证区块时, 若存在分片 A 发起的跨分片 A, B 的交易 (原始交易保存在分片 A), 由于分片 B 仅保存了子交易, 其需要向分片 A 请求原始交易验证, 查询开销较高. 在跨域场景下, 若跨分片交易数量较多或存储空间较少 (如 IoT 领域), 可以使用多子链的方式存储.

为了解决查询跨分片交易造成的高开销, Caper [28], Sharper [42]等系统将账本存储为有向无环图 (DAG)的形式, 每个分片保存了所有涉及该分片的交易 (DAG 的子图), 跨分片交易能在所涉及到的任意分片内均可查询, 查询开销较低. 但是, 这样做也会导致跨分片交易在多个分片内保存有完整的副本, 增加了存储开销. 因此, 若跨域带宽较为宝贵且跨分片查询请求较多, 可以考虑使用 DAG 的方式存储账本.

Caper [28]针对多应用协作设计, 每个应用负责保存该应用的本地交易和所有跨应用交易. 由于 Caper 的多应用协作与跨分片提交协议相同, 本节接下来使用分片替代应用做说明. 在图 13(a) 中, Caper 将每个交易作为一个区块, 四个分片组成的 Caper 区块链账本, $t_{i,j}$ 表示跨分片事务, i 代表所属分片发起的事务顺序, j 代表跨分片的事务顺序, 例如 $t_{23,2}$ 就是第 2, 3 个分片发起的, 系统中第 2 个跨分片事务, 需要由所有分片保存. 而交易 t_{15} 为分片 1 内第 5 个分片内交易, 仅由分片 1 保存. 交易间使用哈希指针连接, 用以确定交易间的执行顺序.

Sharper [42]的账本结构和 Caper 相似. 在四个数据分片组成的系统中, 每个集群也仅维护自己的账本视图, 每个账本分片同样由片内事务和跨分片事务组成. 但是, 不像在 Caper 中跨分片交易需要在所有分片内存储, Sharper 中的跨分片交易只需要在对应分片内保存, 涉及到不同分片的交易可以并发执行. 例如, 在上述例子中, Sharper 只需要分片 2, 3 保存 $t_{23,2}$, 其他分片无需存储 $t_{23,2}$.

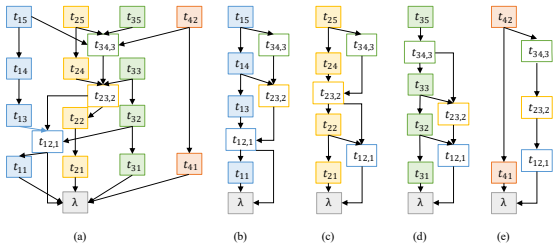

图 13 Caper 的 DAG 账本结构

3.3 分片的放置

大多分片区块链 (如 Elastico、OmniLedger、RapidChain 等) 并未考虑分片数据的放置策略. 由于节点使用无偏随机函数被划分到不同的分片, 当节点部署在多个数据中心时且每个数据中心内节点足够多时, 一个分片会在所有数据中心内保存有副本. 此时, 分片内通信需要跨多个数据中心, 严重影响系统性能. 为了最小化分片内通信的开销, 最理想的情况为让一个分片内的副本存储在相同数据中心. 因为数据中心内网络远高于数据中心间的网络, 这样可以避免网络成为分片内通信的瓶颈.

基于上述思想, Saguaro[96]提出了一种树形结构用于边缘计算领域. 该结构分为四层, 最下层为边缘设备 (edge devices), 如传感器, 车辆等. 在同一城市的边缘设备使用部署在相同城市的一组边缘服务器 (edge servers) 管理, 这组边缘服务器负责存储所管理的边缘设备数据. 类似的, 在每个省部署一组雾服务器 (fog servers), 用于管理该省的所有边缘服务器. 最后, 云服务器 (cloud servers) 管理所有雾服务器. 这样, 区块链数据得以按照城市分片, 并且每个分片放置在对应城市中, 用来最小化用户的访问时间和分片内共识时间. 同时, 为了在数据中心火灾等情况下分片数据不丢失, 树中的非叶子节点延迟同步其叶节点的账本分片. 最后, Saguaro 还支持动态迁移分片的记录 (即按照访问地理位置动态重新划分数据). 当边缘设备转移城市时, 它所对应的记录也可以得到迁移, 尽可能保证数据的本地访问.

Saguaro 的延迟同步算法为异步备份分片, 通过牺牲分片的可用性保证高性能, 当数据中心不可用时, 最新数据会丢失. 与之不同, Ziziphus[20]让数据在数个地理位置相近的数据中心通过共识协议完成复制, 在保证性能的同时提供了数据可用性. 具体而言, Ziziphus 将数据中心 (zone) 按照地理位置划分为集群 (zone cluster), 边缘设备的数据可以在一个或数个集群内使用跨集群数据同步算法 (cross-cluster data

synchronization) 备份. 这样一来, 一个或数个数据中心受灾不会导致数据丢失, 并且因为集群内数据中心地理位置相近, 分片内共识延迟较低.

3.4 分片数据的划分

以上介绍账户 (记录) 的划分策略均为基于账户的哈希值来对确定所属分片, 属于随机划分. 例如, RapidChain[40]利用账户属性哈希将账户分到不同的分片. 这种划分方法是确定性的, 划分账户不会影响系统性能. 但是, 这种方法未考虑到账户间的访问关系, 例如, 用户通常会在一个交易内访问属于同一地域传感器记录, 而不同地域间的传感器记录通常不会在一个交易内联合访问. 因此, 这种不考虑记录间关系的随机方式会引入大量的跨分片交易. 在跨域联盟链中, 处理跨分片交易的代价高, 严重影响系统的性能.

为减少跨分片交易, 基于图结构的划分方法[133]将交易频繁的账户划分到一个分片里. 其中, 账户为图中的点, 涉及到这些账户的交易为边, 账户间交易频率为边权重. 可以利用图分割技术 Metis[118]将原图划分成多个子图, 每个子图代表一个分片. 由于跨子图边的权重和最小, 以此划分得到的分片执行跨分片交易对系统影响最小, 尽可能最大化分片内交易的比率.

但由于 Metis 的划分策略只是将交易频繁而被划分到一个分片内, 各个子图内边的权重和可能相差较大. 以此分片可能导致各个分片内负载不均衡, 导致某个分片过热, 降低系统性能. 为均衡各个分片的负载同时降低跨分片交易比率, TxAllo[125]对账户图结构进行建模, 利用了社区检测算法动态地划分账户所属的分片. 一个分片代表一个社区, 包含了多个紧密相连的账户节点 代表着分片内账户交易频繁. 而社区相互之间稀疏松散, 跨分片的交易数量少. TxAllo 分为两个阶段, (1) 初始化阶段: 社区检测算法自动将交易频繁的账户划分到一个分片, 跨分片的比例较低. (2) 优化阶段: 通过调整账户所处的分片来最大化系统的负载均衡. 每个账户会依次加入不同分片来评估总的吞吐量增益, 增益的计算包含了跨分片交易、吞吐量、负载等多种性能指标. 这种分片技术不仅降低跨分片的交易数量而且根据跨域交易的处理代价动态保证分片的负载均衡.

TxAllo 假设各个分片的处理能力相同. 然而, 在跨域联盟链场景下, 不同的分片可能由不同的组织管理 (如 Qanaat) 或包含不同数量的物理节点, 导致各个分片拥有不同的处理能力. 因此, 可以让 TxAllo 考虑不同分片的处理能力划分账户, 为处理能力强的分片分配更多的账户, 进一步提升系统的性能.

3.5 分片技术总结

表 5 跨分片提交协议和对应区块链系统

区块链系统	年份	提交协议		特征
AHL[41]	2019	中心化 2PC	参考委员会	共识协议使用可信执行环境(TEE), 开销较低
OmniLedger[65]	2018		客户端驱动	客户端面向整个网络广播, 负担较重
ChainSpace[135]	2017		客户端驱动	可以对智能合约进行分片
RapidChain[40]	2018		分片驱动	基于 UTXO 交易性质拆分跨分片交易
GeoChain[29]	2023		客户端驱动	改进的客户端驱动 2PC 协议, 客户端广播分片交易证明
Caper[28]	2019	三种提交协议		面向企业协作, 账本采用 DAG 存储, 企业仅维护相关部分视图
Sharper[42]	2021	分布式 2PC		可以适用于 CFT 和 BFT 两种安全模型
Qanaat[134]	2021	中心化+分布式 2PC		面向企业协作, 相比 Caper 提升系统扩展性和企业数据的隐私性
Monoxide[127]	2019	原子提交协议		跨分片协调开销低, 但只适用于部分类型交易
Meepo[131]	2021	Cross-epoch 和 Cross-call		分片间交换信息来确定事务执行顺序, 减少通信开销
Pyramid[132]	2021	分层提交协议		分层的分片区块链, 多个节点同时处理分片事务
CoChain[124]	2023	原子提交协议等[124]		分片间建立共识组, 允许分片被拜占庭节点攻占
BrokerChain[133]	2022	Broker 账户		基于 Metis 图算法划分分片, 减少跨分片事务数量
Prophet[123]	2023	确定性并发控制		跨分片交易提交结果是确定性的, 不需要进行额外协调
Saguaro[96]	2023	中心化 2PC		分片树形分层账本结构, 根据边缘设备位置跨分片迁移数据
Ziziphus[20]	2023	数据同步+数据迁移协议		以集群为单位同步分片, 在性能和可用性方面取得权衡

表 5 总结了上述系统的提交协议. 中心化 2PC 协议由协调者接收和广播交易信息, 分片内对 2PC 决策进行共识来抵御拜占庭节点, 跨域的通信开销较高. 分布式 2PC 协议为减少跨数据中心的通信次数, 分片间进行全连接广播通信提交交易. 原子提交协议不需要分片间的协调工作, 但仅能用于部分类型的事务; 分层提交协议可以本地验证其中的跨分片交易, 减少通信的次数; 确定性的排序算法避免了跨分片交易间的冲突,

跨分片交易不需要使用传统 2PC 提交; Saguaro 和 Ziziphus 的分片放置策略可以有效减少分片内跨域共识的开销; 基于 Metis 的图划分策略和 TxAllo 账户划分策略也减少了跨分片交易的比率.

4 总结和展望

随着全球经济的迅速发展, 企业的规模不断扩大, 对联盟区块链的需求也日益增加. 为满足更加广泛和大规模的部署需求, 联盟链必须兼具有公有链的良好扩展性, 同时确保在高吞吐量、低延迟和高扩展性方面具备卓越的性能. 然而, 跨地域部署所面临的主要挑战之一是节点间较高的通信延迟. 尽管如此, 随着技术的不断进步, 当前的共识协议、事务处理机制和分片技术已经逐渐能够适应跨地理区域的联盟链部署. 这些技术可以根据跨域网络架构的特性进行调整, 从而优化系统的性能, 提升联盟链在复杂、多变的全球性业务环境中的适应能力. 这种技术策略的调整将有助于联盟链在全球范围内的广泛应用, 为企业提供更加可靠和高效的区块链解决方案. 根据上述需求, 本文对跨域联盟链提出了如下展望:

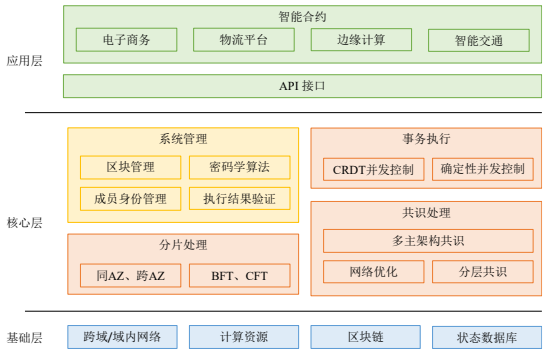

图 14 跨域联盟区块链的层次结构图

图 14 展示了跨域联盟链系统的未来层次结构, 其中基础层为系统提供底层支持, 核心层为系统的性能、可用性、可扩展性提供保证, 并在应用层支持多种跨域高性能应用. 在基础层中, 跨域异构网络和云平台提供的庞大计算资源等硬件特性决定了跨域联盟链的结构. 共识处理、事务执行、分片处理等模块的设计需要充分考虑到基础层的硬件特性. 为保证不可篡改、透明性、可溯源等特性, 系统底层仍需采用区块链的链式存储结构, 但可以使用如 LevelDB、RocksDB 等高性能键值数据库缓存状态信息来高效执行交易.

对于核心层的设计与实现, 本文已经对跨域共识协议、并发执行技术以及区块链分片技术进行了深入分析. 在综合这些技术跨地域部署时的优势与局限后, 本文提出了如下创新的范式:

在共识协议方面, 为利用跨域基础设施, 应使用 GeoBFT、Steward 等协议提出的分层共识架构. 域内可以使用 PBFT、HotStuff 等共识协议防止拜占庭容错, 而域间根据不同的可用性需求, 可以使用全复制 (如 Steward) 容忍不超过半数数据中心宕机, 也可以使用 Ziziphus 等系统的部分复制模型, 仅容忍数个数据中心宕机换取更好的性能. 通信优化上, 因为跨域公网带宽宝贵, 数据中心集群间交换区块可以使用纠删码编码区块后传输, 有效降低冗余传输. 还可以利用 Gossip 协议或 RS-Paxos 等算法计算最优编码方式和传输策略, 结合门限签名等技术, 进一步降低区块传输开销. 域内共识可使用 Infiniband、可编程交换机等硬件, 而域间区块传输则可以使用流水线技术来进一步提升吞吐率.

在事务处理方面, 跨域联盟链系统节点间存在高的通信延迟, 减少节点间通信次数尤为重要. 在要求强隔离级别场景下, 确定性并发控制更适合跨域联盟链, 它仅共识交易输入, 节点间通信次数少, 网络开销较低, 节点在收到交易后按照确定性的顺序执行得到相同的结果, 执行过程不需要相互通信 (特指非分片情况). 而如果工作负载不需要强隔离界别, 无协调一致性技术更适合跨域联盟链, 节点间无需进行共识通信, 可以按照任意顺序执行 CRDT 交易均会得到一致的结果, 能够更好地利用跨域带宽和节点的计算资源.

在分片方面, 跨域联盟链应充分考虑分片的副本集群间的地理位置关系和跨分片交易数量. 由于跨域的网络通信高, 分片提交协议应避免多次的协调通信, Prophet 的确定性排序分片架构可以避免跨分片交易冲突导致的低吞吐量, 适合于跨域联盟链. 而基于图结构的分片划分技术在减少跨域跨分片的交易同时应该均衡好各个分片的工作负载。跨域联盟链还可以应用 TxAllo 的算法, 将跨域的通信代价加入到吞吐量增益计算中, 平衡各个分片在处理跨域下的工作负载, 提升系统性能.

在系统应用层中, 系统可以通过智能合约等方式支持电子商务、物流平台、边缘计算、智能交通等大规模跨数据中心的应用. 同时针对不同的应用场景设计出对应的 API 接口, 满足每个应用的使用. 例如, 针对电子商务提供支付接口等. 通过基础层和核心层的技术来保证在大规模跨数据中心应用的安全性和可用性.

本文介绍了跨地理区域联盟链的架构和工作原理, 详细介绍了现有区块链系统针对跨域部署提供的解决方案. 通过梳理了每个系统的特点, 归纳并分析了不同的系统的技术方法, 并对跨数据中心的区块链系统研究方向提出了总结和展望, 希望为未来的发展提供了有益的指导.

References:

[1] Beck R, Müller-Bloch C, King JL. Governance in the blockchain economy: A framework and research agenda. Journal of the association for information systems, 2018,19(10):1.

[2] Böhme R, Christin N, Edelman B, Moore T. Bitcoin: Economics, technology, and governance. Journal of economic Perspectives, 2015,29(2):213-238.

[3] Huckle S, Bhattacharya R, White M, Beloff N. Internet of Things, Blockchain and Shared Economy Applications. 7th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (Euspn 2016)/the 6th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (Icth-2016), 2016,98461-466.

[4] Swan M. Blockchain: Blueprint for a new economy[M]. " O'Reilly Media, Inc.", 2015.

[5] Liu C, Xiao Y, Javangula V, Hu Q, Wang S, Cheng X. NormaChain: A blockchain-based normalized autonomous transaction settlement system for IoT-based E-commerce. IEEE Internet of Things Journal, 2018,6(3):4680-4693.

[6] Liu Z, Li Z. A blockchain-based framework of cross-border e-commerce supply chain. International journal of information management, 2020,52102059.

[7] Dorri A, Kanhere SS, Jurdak R. Towards an optimized blockchain for IoT. In: Proc. of the second international conference on Internet-of-Things design and implementation. 2017: 173-178.

[8] Reyna A, Martín C, Chen J, Soler E, Díaz M. On blockchain and its integration with IoT. Challenges and opportunities. Future generation computer systems, 2018,88173-190.

[9] Wang Q, Zhu X, Ni Y, Gu L, Zhu H. Blockchain for the IoT and industrial IoT: A review. Internet of Things, 2020,10100081.

[10] Azaria A, Ekblaw A, Vieira T, Lippman A. Medrec: Using blockchain for medical data access and permission management. In: Proc. of the 2016 2nd International Conf on Open and Big Data. Vienna: IEEE, 2016: 25-30.

[11] Chen Y, Ding S, Xu Z, Zheng H, Yang S. Blockchain-based medical records secure storage and medical service framework. Journal of medical systems, 2019,431-9.

[12] Fan K, Wang S, Ren Y, Li H, Yang Y. Medblock: Efficient and secure medical data sharing via blockchain. Journal of medical systems, 2018,421-11.

[13] Astarita V, Giofrè VP, Mirabelli G, Solina V. A review of blockchain-based systems in transportation. Information, 2019,11(1):21.

[14] Mollah MB, Zhao J, Niyato D, Guan YL, Yuen C, Sun S, Lam K-Y, Koh LH. Blockchain for the internet of vehicles towards intelligent transportation systems: A survey. IEEE Internet of Things Journal, 2020,8(6):4157-4185.

[15] Yuan Y, Wang F-Y. Towards blockchain-based intelligent transportation systems. In: Proc. of the 2016 IEEE 19th International Conf on Intelligent Transportation Systems. Rio de Janeiro: IEEE, 2016: 2663-2668.

[16] Zhongguancun Blockchain Industry Alliance. Global Blockchain Industry Map Report, 2024-03-01, https://mp.weixin.qq.com/s/AKi0rrGJzAniWaeu_-WdIw. (in Chinese with English abstract).

[17] Xu Shoufang, Zhang Jianlin. Construction of "Belt and Road" Cross-border Logistics Platform Based on Blockchain Technology.Wu Liu Ji Shu/Logistics Technology, 2018, 37(7): 56-61+124. (in Chinese with English abstract)

[18] Zhou Fuli, Hai Panpan, Chen Tianfu, He Yandong, Chen Shan. Research on The Integration Mechanism and Realization Path of Blockchain Technology and Cross-border E-commerce.World Science and Technology Research and Development, 2023, 45(1): 41-50. (in Chinese with English abstract)
[19] Inghirami IE. Accounting information systems in the time of blockchain. In: Proc. of the İtaıs 2018 Conf. 2019: 1-16.
[20] Amiri MJ, Shu D, Maiyya S, Agrawal D, El Abbadi A. Ziziphus: Scalable data management across byzantine edge servers. In: Proc. of the 2023 IEEE 39th International Conf on Data Engineering. Anaheim: IEEE, 2023: 490-502.
[21] Jayatunga E, Nag A, Jurcut AD. Security Requirements for Vehicle-to-Everything (V2X) Communications Integrated with Blockchain. In: Proc. of the 2022 Fourth International Conf on Blockchain Computing and Applications. San Antonio: IEEE, 2022: 208-213.
[22] Taft R, Sharif I, Matei A, VanBenschoten N, Lewis J, Grieger T, Niemi K, Woods A, Birzin A, Poss R. Cockroachdb: The resilient geo-distributed sql database. In: Proc. of the 2020 ACM SIGMOD International Conf on Management of Data. New York: ACM, 2020: 1493-1509.
[23] Huawei, global-infrastructure Huawei Cloud, 2024-03-01, https://www.huaweicloud.com/intl/en-us/about/global-infrastructure.html. (in Chinese with English abstract).
[24] Alibaba Cloud Documentation, Alibaba Cloud Documentation, 2023, https://help.aliyun.com/
[25] Wu CG, Faleiro JM, Lin YH, Hellerstein JM. Anna: A kvs for any scale. IEEE Trans. on Knowledge and Data Engineering, 2019,33(2):344-358.
[26] de Arruda EH, Lunardi RC, Nunes HC, Zorzo AF, Michelin RA. Appendable-block blockchain evaluation over geographically-distributed IoT networks. In: Proc. of the 2020 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom). IEEE, 2020: 1-6.
[27] Li H, Chen Y, Shi X, Bai X, Mo N, Li W, Guo R, Wang Z, Sun Y. Fisco-bcos: An enterprise-grade permissioned blockchain system with high-performance. In: Proc. of International Conference for High Performance Computing, Networking, Storage and Analysis. 2023: 1-17.
[28] Amiri MJ, Agrawal D, El Abbadi A. CAPER: A Cross-Application Permissioned Blockchain. In: Proc. of the VLDB Endowment, 2019,12(11):1385-1398.
[29] Mao CY, Golab W. Geochain: A locality-based sharding protocol for permissioned blockchains. In: Proc. of the 24th International Conf on Distributed Computing and Networking. New York: ACM, 2023: 70-79.
[30] Moniz H, Leitao J, Dias RJ, Gehrke J, Preguica N, Rodrigues R. Blotter: Low Latency Transactions for Geo-Replicated Storage. In: Proc. of the 26th International Conf on World Wide Web. Republic and Canton of Geneva: International World Wide Web Conferences Steering Committee, 2017: 263-272, 2017.[doi: 10.1145/3038912.3052603].
[31] Gupta S, Rahnama S, Hellings J, Sadoghi M. ResilientDB: Global Scale Resilient Blockchain Fabric. In: Proc. of the VLDB Endowment, 2020,13(6):868-883.
[32] Androulaki E, Barger A, Bortnikov V, Cachin C, Christidis K, De Caro A, Enyeart D, Ferris C, Laventman G, Manevich Y, Muralidharan S, Murthy C, Binh N, Sethi M, Singh G, Smith K, Sorniotti A, Stathakopoulou C, Vukolic M, Cocco SW, Yellick J, Assoc Comp M. Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. In: Proc. of the 13th EuroSys Conf. New York: ACM, 2018, 2018.[doi: 10.1145/3190508.3190538].
[33] Chase JPM, A Permissioned Implementation of Ethereum, 2024-03-01, https://github.com/jpmorganchase/quorum
[34] Castro M, Liskov B. Practical byzantine fault tolerance and proactive recovery. Acm Transactions on Computer Systems, 2002,20(4):398-461.
[35] Ongaro D, Ousterhout J. The raft consensus algorithm. Lecture Notes CS, 2015,1902022.
[36] Bitcoin NS. Bitcoin: A peer-to-peer electronic cash system. 2008.
[37] Sharding in Ethereum 2.0, 2024-03-01, https://ethereum.org/en/upgrades/sharding/
[38] Peng Z, Zhang Y, Xu Q, Liu H, Gao Y, Li X, Yu G. Neuchain: a fast permissioned blockchain system with deterministic ordering. In: Proc. of the VLDB Endowment, 2022,15(11):2585-2598.
[39] Nasirifard P, Mayer R, Jacobsen H-A. OrderlessChain: a CRDT-enabled blockchain without total global order of transactions. In: Proc. of the 23rd International Middleware Conf Demos and Posters. New York: ACM, 2022: 5-6.
[40] Zamani M, Movahedi M, Raykova M. Rapidchain: Scaling blockchain via full sharding. In: Proc. of the 2018 ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2018: 931-948.
[41] Dang H, Dinh TTA, Loghin D, Chang E-C, Lin Q, Ooi BC. Towards scaling blockchain systems via sharding. In: Proc. of the 2019 International Conf on Management of Data. New York: ACM, 2019: 123-140.

[42] Amiri MJ, Agrawal D, El Abbadi A. Sharper: Sharding permissioned blockchains over network clusters. In: Proc. of the 2021 International Conf on Management of Data. New York: ACM, 2021: 76-88.

[43] Hong ZC, Guo S, Zhou EY, Chen WH, Huang HW, Zomaya A. GriDB: Scaling Blockchain Database via Sharding and Off-Chain Cross-Shard Mechanism. In: Proc. of the VLDB Endowment, 2023,16(7):1685-1698.

[44] Hellings J, Sadoghi M. Byshard: Sharding in a byzantine environment. The VLDB Journal, 2023,32(6):1343-1367.

[45] Berger C, Schwarz-Rüsch S, Vogel A, Bleeke K, Jehl L, Reiser HP, Kapitza R. Sok: Scalability techniques for BFT consensus. In: Proc. of the 2023 IEEE International Conference on Blockchain and Cryptocurrency (ICBC). IEEE, 2023: 1-18.

[46] Zhou QH, Huang HW, Zheng ZB, Bian J. Solutions to Scalability of Blockchain: A Survey. Ieee Access, 2020,816440-16455.

[47] Yu GS, Wang X, Yu K, Ni W, Zhang JA, Liu RP. Survey: Sharding in Blockchains. Ieee Access, 2020,814155-14181.

[48] Conti M, Kumar G, Nerurkar P, Saha R, Vigneri L. A survey on security challenges and solutions in the IOTA. Journal of Network and Computer Applications, 2022,203103383.

[49] Lamport L. Paxos made simple. ACM SIGACT News (Distributed Computing Column) 32, 4 (Whole Number 121, December 2001), 2001,51-58.

[50] Amir Y, Danilov C, Dolev D, Kirsch J, Lane J, Nita-Rotaru C, Olsen J, Zage D. Steward: Scaling byzantine fault-tolerant replication to wide area networks. IEEE Trans. on Dependable and Secure Computing, 2008,7(1):80-93.

[51] Charapko A, Ailijiang A, Demirbas M. Pigpaxos: Devouring the communication bottlenecks in distributed consensus. In: Proc. of the 2021 International Conf on Management of Data. New York: ACM, 2021: 235-247.

[52] Guo Bing Yong, Lu Zhen Liang, Tang Qiang, Xu Jing, Zhang Zhen Feng. Dumbo: Faster asynchronous bft protocols. In: Proc. of the 2020 ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2020: 803-818.

[53] Miller A, Xia Y, Croman K, Shi E, Song D. The honey badger of BFT protocols. In: Proc. of the 2016 ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2016: 31-42.

[54] Nawab F, Sadoghi M. Blockplane: A global-scale byzantizing middleware. In: Proc. of the 2019 IEEE 35th International Conf on Data Engineering. Macao: IEEE, 2019: 124-135.

[55] Liu FG, Yang YY. D-Paxos: Building Hierarchical Replicated State Machine for Cloud Environments. Ieice Transactions on Information and Systems, 2016,E99d(6):1485-1501.

[56] Castiglia T, Goldberg C, Patterson S. A hierarchical model for fast distributed consensus in dynamic networks. In: Proc. of the 2020 IEEE 40th International Conference on Distributed Computing Systems. Singapore: IEEE, 2020: 1189-1190.

[57] Van Renesse R, Altinbuken D. Paxos made moderately complex. ACM Computing Surveys, 2015,47(3):1-36.

[58] Rizvi S, Wong B, Keshav S. Canopus: A scalable and massively parallel consensus protocol. In: Proc. of the 13th International Conf on Emerging Networking EXperiments and Technologies. New York: ACM, 2017: 426-438.

[59] Dwork C, Lynch N, Stockmeyer L. Consensus in the presence of partial synchrony. Journal of the ACM, 1988,35(2):288-323.

[60] Keshav S, Golab W, Wong B, Rizvi S, Gorbunov S. RCanopus: Making canopus resilient to failures and byzantine faults. arXiv Preprint arXiv:181009300, 2018.

[61] Neiheiser R, Matos M, Rodrigues L. Kauri: Scalable bft consensus with pipelined tree-based dissemination and aggregation. In: Proc. of the ACM SIGOPS 28th Symposium on Operating Systems Principles. New York: ACM, 2021: 35-48.

[62] Yin MF, Malkhi D, Reiter MK, Gueta GG, Abraham I. HotStuff: BFT consensus with linearity and responsiveness. In: Proc. of the 2019 ACM Symposium on Principles of Distributed Computing. New York: ACM, 2019: 347-356.

[63] Kogias EK, Jovanovic P, Gailly N, Khoffi I, Gasser L, Ford B. Enhancing bitcoin security and performance with strong consistency via collective signing. In: Proc. of the 25th USENIX Security Symposium. Vancouver: USENIX, 2016: 279-296.

[64] Syta E, Tamas I, Visher D, Wolinsky DI, Jovanovic P, Gasser L, Gailly N, Khoffi I, Ford B. Keeping authorities" honest or bust" with decentralized witness cosigning. In: Proc. of the 2016 IEEE Symposium on Security and Privacy. San Jose: IEEE, 2016: 526-545.

[65] Kokoris-Kogias E, Jovanovic P, Gasser L, Gailly N, Syta E, Ford B. Omniledger: A secure, scale-out, decentralized ledger via sharding. In: Proc. of the 2018 IEEE Symposium on Security and Privacy. San Francisco: IEEE, 2018: 583-598.

[66] Syta E, Jovanovic P, Kogias EK, Gailly N, Gasser L, Khoffi I, Fischer MJ, Ford B. Scalable bias-resistant distributed randomness. In: Proc. of the 2017 IEEE Symposium on Security and Privacy. San Jose: IEEE, 2017: 444-460.

[67] Micali S, Rabin M, Vadhan S. Verifiable random functions. In: Proc. of the 40th Annual Symposium on Foundations of Computer Science. New York: IEEE, 1999: 120-130.
[68] Lu Y, Yu X, Cao L, Madden S. Aria: a fast and practical deterministic OLTP database. 2020.
[69] Arun B, Ravindran B. Scalable Byzantine Fault Tolerance via Partial Decentralization. In: Proc. of the VLDB Endowment, 2022,15(9):1739-1752.
[70] Fischer MJ, Lynch NA, Paterson MS. Impossibility of distributed consensus with one faulty process. Journal of the ACM, 1985,32(2):374-382.
[71] Mostéfaoui A, Moumen H, Raynal M. Signature-free asynchronous Byzantine consensus with t< n/3 and O (n2) messages. In: Proc. of the 2014 ACM Symposium on Principles of Distributed Computing. New York: ACM, 2014: 2-9.
[72] Ben-Or M, Kelmer B, Rabin T. Asynchronous secure computations with optimal resilience. In: Proc. of the 13th Annual ACM Symposium on Principles of Distributed Computing. New York: ACM, 1994: 183-192.
[73] Bracha G. Asynchronous Byzantine agreement protocols. Information and Computation, 1987,75(2):130-143.
[74] Shoup V. Practical threshold signatures. In: Proc. of the International Conf on the Theory and Application of Cryptographic Techniques. Bruges: Springer, 2000: 207-220.
[75] Desmedt Y. Threshold cryptosystems. In: Proc. of the International Workshop on the Theory and Application of Cryptographic Techniques. Queensland: Springer, 1992: 1-14.
[76] Boneh D, Lynn B, Shacham H. Short signatures from the Weil pairing. In: Proc. of the International Conf on the Theory and Application of Cryptology and Information Security. Gold Coast: Springer, 2001: 514-532.
[77] Baek J, Zheng YL. Simple and efficient threshold cryptosystem from the gap diffie-hellman group. In: Proc. of the IEEE Global Telecommunications Conf. IEEE, 2003: 1491-1495.
[78] Ben-Or M, El-Yaniv R. Resilient-optimal interactive consistency in constant time. Distributed Computing, 2003,16(4):249-262.
[79] Liu SY, Xu WB, Shan C, Yan XF, Xu TJ, Wang B, Fan L, Deng FX, Yan Y, Zhang H. Flexible Advancement in Asynchronous BFT Consensus. In: Proc. of the 29th Symposium on Operating Systems Principles. New York: ACM, 2023: 264-280.
[80] Poke M, Hoefler T. Dare: High-performance state machine replication on rdma networks. In: Proc. of the 24th International Symposium on High-Performance Parallel and Distributed Computing. New York: ACM, 2015: 107-118.
[81] Wang C, Jiang JY, Chen XS, Yi N, Cui HM. Apus: Fast and scalable paxos on rdma. In: Proc. of the 2017 Symposium on Cloud Computing. New York: ACM, 2017: 94-107.
[82] Wu CY, Amiri MJ, Asch J, Nagda H, Zhang QZ, Loo BT. FlexChain: An Elastic Disaggregated Blockchain. In: Proc. of the VLDB Endowment, 2022,16(1):23-36.
[83] Li JL, Michael E, Sharma NK, Szekeres A, Ports DR. Just say {NO} to paxos overhead: Replacing consensus with network ordering. In: Proc. of the 12th USENIX Symposium on Operating Systems Design and Implementation. Savannah: USENIX, 2016: 467-483.
[84] Massey J. Theory and practice of error control codes. In: Proc. of the IEEE, 1986,74(9):1293-1294.
[85] Reed IS, Solomon G. Polynomial codes over certain finite fields. Journal of the Society for Industrial and Applied Mathematics, 1960,8(2):300-304.
[86] Mu S, Chen K, Wu YW, Zheng WM. When paxos meets erasure code: Reduce network and storage cost in state machine replication. In: Proc. of the 23rd International Symposium on High-performance Parallel and Distributed Computing. New York: ACM, 2014: 61-72.
[87] Xu MW, Zhou Y, Qiao YY, Xu K, Wang Y, Yang J. ECRaft: A Raft Based Consensus Protocol for Highly Available and Reliable Erasure-Coded Storage Systems. In: Proc. of the 2021 IEEE 27th International Conference on Parallel and Distributed Systems. Beijing: IEEE, 2021: 707-714.
[88] Jia YL, Xu GP, Sung CW, Mostafa S, Wu YL. HRaft: Adaptive Erasure Coded Data Maintenance for Consensus in Distributed Networks. In: Proc. of the 2022 IEEE International Parallel and Distributed Processing Symposium. Lyon: IEEE, 2022: 1316-1326.
[89] Zhang M, Kang QH, Lee PP. Minimizing Network and Storage Costs for Consensus with Flexible Erasure Coding. In: Proc. of the 52nd International Conf on Parallel Processing. New York: ACM, 2023: 41-50.
[90] Kaklamanis I, Yang L, Alizadeh M. Poster: Coded Broadcast for Scalable Leader-Based BFT Consensus. In: Proc. of the 2022 ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2022: 3375-3377.

[91] Chawla N, Behrens HW, Tapp D, Boscovic D, Candan KS. Velocity: Scalability improvements in block propagation through rateless erasure coding. In: Proc. of the 2019 IEEE International Conf on Blockchain and Cryptocurrency. Seoul: IEEE, 2019: 447-454.

[92] Hellings J, Sadoghi M. Coordination-free byzantine replication with minimal communication costs. In: Proc. of the 23rd International Conference on Database Theory. Copenhagen: Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

[93] Cachin C, Tessaro S. Asynchronous verifiable information dispersal. In: Proc. of the 24th IEEE Symposium on Reliable Distributed Systems. Orlando: IEEE, 2005: 191-201.

[94] Stathakopoulou C, Pavlovic M, Vukolić M. State machine replication scalability made simple. In: Proc. of the Seventeenth European Conf on Computer Systems. New York: ACM, 2022: 17-33.

[95] Gueta GG, Abraham I, Grossman S, Malkhi D, Pinkas B, Reiter M, Seredinschi D-A, Tamir O, Tomescu A. SBFT: A scalable and decentralized trust infrastructure. In: Proc. of the 2019 49th Annual IEEE/IFIP International Conf on Dependable Systems and Networks. Portland: IEEE, 2019: 568-580.

[96] Amiri MJ, Lai ZL, Patel L, Loo BT, Lo E, Zhou WC. Saguaro: An edge computing-enabled hierarchical permissioned blockchain. In: Proc. of the 2023 IEEE 39th International Conf on Data Engineering. Anaheim: IEEE, 2023: 259-272.

[97] Gupta S, Hellings J, Rahnama S, Sadoghi M. Proof-of-execution: Reaching consensus through fault-tolerant speculation. arXiv Preprint arXiv:191100838, 2019.

[98] Nick J, Ruffing T, Seurin Y. MuSig2: simple two-round Schnorr multi-signatures. In: Proc. of the Annual International Cryptology Conf. New York: Springer, 2021: 189-221.

[99] Sousa J, Bessani A. From Byzantine consensus to BFT state machine replication: A latency-optimal transformation. In: Proc. of the 2012 Ninth European Dependable Computing Conf. Sibiu: IEEE, 2012: 37-48.

[100] Gupta S, Hellings J, Sadoghi M. RCC: resilient concurrent consensus for high-throughput secure transaction processing. In: Proc. of the 2021 IEEE 37th International Conf on Data Engineering. Chania: IEEE, 2021: 1392-1403.

[101] Cao W, Liu ZJ, Wang P, Chen S, Zhu CF, Zheng S, Wang YH, Ma GQ. PolarFS: An Ultra-low Latency and Failure Resilient Distributed File System for Shared Storage Cloud Database. In: Proc. of the VLDB Endowment, 2018,11(12):1849-1862.

[102] Gorenflo C, Lee S, Golab L, Keshav S. FastFabric: Scaling hyperledger fabric to 20 000 transactions per second. International Journal of Network Management, 2020,30(5):e2099.

[103] Qi J, Chen XS, Jiang YP, Jiang JY, Shen TX, Zhao SX, Wang S, Zhang G, Chen L, Au MH. Bidl: A high-throughput, low-latency permissioned blockchain framework for datacenter networks. In: Proc. of the ACM SIGOPS 28th Symposium on Operating Systems Principles. New York: ACM, 2021: 18-34.

[104] Nathan S, Govindarajan C, Saraf A, Sethi M, Jayachandran P. Blockchain meets database: Design and implementation of a blockchain relational database. arXiv preprint arXiv:190301919, 2019.

[105] Ruan PC, Loghin D, Ta Q-T, Zhang MH, Chen G, Ooi BC. A transactional perspective on execute-order-validate blockchains. In: Proc. of the 2020 ACM SIGMOD International Conf on Management of Data. New York: ACM, 2020: 543-557.

[106] Xu C, Zhang C, Xu J, Pei J. SlimChain: Scaling blockchain transactions through off-chain storage and parallel processing. In: Proc. of the VLDB Endowment, 2021,14(11):2314-2326.

[107] Faleiro JM, Abadi DJ. Rethinking serializable multiversion concurrency control. arXiv Preprint arXiv:14122324, 2014.

[108] Amiri MJ, Agrawal D, El Abbadi A. Parblockchain: Leveraging transaction parallelism in permissioned blockchain systems. In: Proc. of the 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS). IEEE, 2019: 1337-1347.

[109] Baheti S, Anjana PS, Peri S, Simmhan Y. DiPETrans: A framework for distributed parallel execution of transactions of blocks in blockchains. Concurrency and Computation: Practice and Experience, 2022,34(10):e6804.

[110] Thomson A, Diamond T, Weng S-C, Ren K, Shao P, Abadi DJ. Calvin: fast distributed transactions for partitioned database systems. In: Proc. of the 2012 ACM SIGMOD international conference on management of data. 2012: 1-12.

[111] Cahill MJ, Röhm U, Fekete AD. Serializable isolation for snapshot databases. ACM Transactions on Database Systems (TODS), 2009,34(4):1-42.

[112] Ports DR, Grittner K. Serializable snapshot isolation in PostgreSQL. arXiv preprint arXiv:12084179, 2012.

[113] Li Z, Romano P, Van Roy P. Sparkle: speculative deterministic concurrency control for partially replicated transactional stores. In: Proc. of the 2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). IEEE, 2019: 164-175.

[114] Faleiro JM, Abadi DJ, Hellerstein JM. High performance transactions via early write visibility. In: Proc. of the VLDB Endowment, 2017,10(5).
[115] Lai Z, Liu C, Lo E. When Private Blockchain Meets Deterministic Database. In: Proc. of the ACM on Management of Data, 2023,1(1):1-28.
[116] Qi XD, Chen ZH, Zhuo HZ, Xu QQ, Zhu CY, Zhang Z, Jin CQ, Zhou AY, Yan Y, Zhang H. SChain: Scalable Concurrency over Flexible Permissioned Blockchain. In: Proc. of the 2023 IEEE 39th International Conf on Data Engineering. Anaheim: IEEE, 2023: 1901-1913.
[117] Chen Z, Zhuo H, Xu Q, Qi X, Zhu C, Zhang Z, Jin C, Zhou A, Yan Y, Zhang H. SChain: a scalable consortium blockchain exploiting intra-and inter-block concurrency. In: Proc. of the VLDB Endowment, 2021,14(12):2799-2802.
[118] Karypis G, Kumar V. METIS: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices. 1998.
[119] Bailis P, Davidson A, Fekete A, Ghodsi A, Hellerstein JM, Stoica I. Highly available transactions: virtues and limitations (extended version). arXiv Preprint arXiv:13020309, 2013.
[120] Shapiro M, Preguiça N, Baquero C, Zawirski M. Conflict-free replicated data types. In: Proc. of the Stabilization, Safety, and Security of Distributed Systems: 13th International Symposium. Grenoble: Springer, 2011: 386-400.
[121] Nasirifard P, Mayer R, Jacobsen H-A. FabricCRDT: A conflict-free replicated datatypes approach to permissioned blockchains. In: Proc. of the 20th International Middleware Conf. New York: ACM, 2019: 110-122.
[122] Sharma A, Schuhknecht FM, Agrawal D, Dittrich J. Blurring the lines between blockchains and database systems: the case of hyperledger fabric. In: Proc. of the 2019 International Conf on Management of Data. New York: ACM, 2019: 105-122.
[123] Hong ZC, Guo S, Zhou EY, Zhang JY, Chen WH, Liang JW, Zhang J, Zomaya A. Prophet: Conflict-Free Sharding Blockchain via Byzantine-Tolerant Deterministic Ordering. arXiv preprint arXiv:230408595, 2023.
[124] Li MZ, Lin Y, Zhang J, Wang W. CoChain: High Concurrency Blockchain Sharding via Consensus on Consensus. In: Proc. of the IEEE INFOCOM 2023-IEEE Conf on Computer Communications. New York: IEEE, 2023: 1-10.
[125] Zhang YZ, Pan SR, Yu JS. Txallo: Dynamic transaction allocation in sharded blockchain systems. In: Proc. of the 2023 IEEE 39th International Conference on Data Engineering. IEEE, 2023: 721-733.
[126] Luu L, Narayanan V, Zheng CD, Baweja K, Gilbert S, Saxena P. A secure sharding protocol for open blockchains. In: Proc. of the 2016 ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2016: 17-30.
[127] Wang JP, Wang H. Monoxide: Scale out blockchains with asynchronous consensus zones. In: Proc. of the 16th USENIX Symposium on Networked Systems Design and Implementation. Boston: USENIX, 2019: 95-112.
[128] Serguei Popov, The tangle. IOTA White paper ver.1.4.3, 2024-03-01, https://www.allcryptowhitepapers.com/iota-whitepaper/
[129] Divya P, Rajeshwaran S, Parthiban R, Ananth Kumar T, Jayalakshmi S. Hashgraph: A Decentralized Security Approach Based on Blockchain with NFT Transactions. In: Proc. of the International Con on Innovations in Data Analytics. West Bengal: Springer, 2022: 33-50.
[130] Fu X, Wang Hi, Shi PC, Ouyang X, Zhang XH. Jointgraph: A DAG‐based efficient consensus algorithm for consortium blockchains. Software: Practice and Experience, 2021,51(10):1987-1999.
[131] Zheng PL, Xu QQ, Zheng ZB, Zhou ZY, Yan Y, Zhang H. Meepo: Sharded consortium blockchain. In: Proc. of the 2021 IEEE 37th International Conf on Data Engineering (ICDE). Chania: IEEE, 2021: 1847-1852.
[132] Hong ZC, Guo S, Li P, Chen WH. Pyramid: A layered sharding blockchain system. In: Proc. of the IEEE INFOCOM 2021-IEEE Conference on Computer Communications. Vancouver: IEEE, 2021: 1-10.
[133] Huang H, Peng X, Zhan J, Zhang S, Lin Y, Zheng Z, Guo S. Brokerchain: A cross-shard blockchain protocol for account/balance-based state sharding. In: Proc. of the IEEE INFOCOM 2022-IEEE Conference on Computer Communications. IEEE, 2022: 1968-1977.
[134] Amiri MJ, Loo BT, Agrawal D, Abbadi AE. Qanaat: A scalable multi-enterprise permissioned blockchain system with confidentiality guarantees. arXiv Preprint arXiv:210710836, 2021.
[135] Al-Bassam M, Sonnino A, Bano S, Hrycyszyn D, Danezis G. Chainspace: A sharded smart contracts platform. arXiv Preprint arXiv:170803778, 2017.

附中文参考文献:

[16] 中关村区块链产业联盟. 全球区块链产业图谱报告. 2024. https://mp.weixin.qq.com/s/AKi0rrGJzAniWaeu_-WdIw

[17] 徐寿芳,章剑林.基于区块链技术的“一带一路”跨境物流平台构建.物流技术,2018,37(07):56-61

[18] 周福礼,海盼盼,陈天赋,何彦东,陈山.区块链技术与跨境电子商务的融合机理与实现路径研究.世界科技研究与发展,2023,45(01):41-50

[23] 华为. 华为借助云计算和人工只能的数字化转型实践.2024. https://e.huawei.com/topic/huawei-own-digital-transformation/cn/index.html

彭泽顺(1997－), 男, 博士生, CCF 学生会员, 主要研究领域为区块链系统, 分布式数据库.

韩智博(2001－), 男, 硕士生, CCF 学生会员, 主要研究领域为区块链系统.

张岩峰(1982－), 男, 博士, 教授, 博士生导师, CCF 高级会员, 主要研究领域为数据库, 大数据处理, 分布式系统.

李晓华(1969－), 女, 博士, 副教授, 硕士生导师, CCF 专业会员, 主要研究领域为信息安全和隐私保护, 移动数据管理, 区块链系统.

于明鹤(1989－), 女, 博士, 讲师, CCF 专业会员, 主要研究领域为数据库, 信息检索.

范吉立(1987－), 男, 硕士, 高级工程师, 主要研究领域为区块链技术, 云计算技术.

于戈(1962－), 男, 博士, 教授, 博士生导师, CCF 会士, 主要研究领域为数据库, 分布式系统.